Nexacro Platform 14 / Administrator Guide

14.0.0.700

14.0.0.700

Nexacro Platform 14 / Administrator Guide

TOBESOFT

TOBESOFT strives to maintain the accuracy of information provided in this document and will consistently modify and supplement it. However, we cannot guarantee accuracy of information in this document. TOBESOFT assumes no responsibility for errors and omissions. No liability is assumed for damages resulting from the use of information contained herein.

Users of this document implicitly agree that they understand the contents of this disclaimer.

Each trademark herein, including the product name of each company, is a solely registered trademark of each company who developed it, and is protected by patent laws and the copyright laws, and should be used for the purpose of reference only.

TOBESOFT Co.,Ltd.
2-5F INTOPS Building, 617 Bongeunsa-Ro, Gangnam-Gu,
Seoul 06083, Korea
www.tobesoft.com

Table of Contents

Copyright and Disclaimer (TOBESOFT) ..... **iv**

Table of Contents ..... **v**

Preface ..... **xi**

Part I. Outline 1

1. Nexacro Platform Introduction **2**

1.1 The Evolution of UI/UX for Business 2

BUX: Business User eXperience 3

1.2 Nexacro Platform 4

Key Features 4

2. Nexacro Platform Outline **6**

2.1 Nexacro Platform Unified Framework 6

2.2 Nexacro Platform Components 8

Programming language 8

Development Environment 10

Implementation Environment 11

The Run-time Environment 12

2.3 System requirements 14

3. Nexacro Platform Distribution **16**

3.1 Distribution Overview 16

3.2 Distribution Files 17

Runtime for Windows 17

HTML5 18

RUNTIME for iOS/Android 19

Part II. Runtime for Windows 20

4. Outline of Nexacro Platform Distribution **21**

4.1 Phases of Distribution 21

4.2 Distribution Flow ... 22
Phase 1 : Nexacro Platform Engine Setup ... 22
Phase 2 : Setting Nexacro Platform Application Program Execute Environment ... 22
Phase 3 : Application Program Execution ... 23
4.3 Setup Files ... 24
Modules ... 24
License files ... 24
Functions of Main Modules ... 25
4.4 Types of Nexacro Platform Distributions ... 26
Runtime Execution Form ... 26
Types of Basic Distribution ... 27
4.5 Version Administration ... 29
Engine Upgrade Information Registration ... 29
Executing Nexacro Platform Engine Upgrade ... 30

Part III. Runtime for iOS/Android ... 31

5. App Development Outline ... 32
5.1 Technology Components ... 32
Device APIs ... 32
Development ... 34
Execution ... 35
Distribution Environment ... 36
5.2 Development Process ... 37
Major Components ... 37
Development and Execution Process ... 39
Distribution Process and Method ... 39
5.3 Specifications for App Development and Execution ... 41
Android ... 41
iOS ... 41
6. Packaging ... 43
6.1 Base Libraries ... 44
6.2 Archive: Make Application Archive ... 45
6.3 Archive: Make Bootstrap ... 48
6.4 Archive: Make File Archive ... 53
6.5 Archive: Make Theme Archive ... 57
6.6 Archive: Package Builder ... 60
7. App Development and Execution (iOS) ... 61
7.1 App Development Environment Configuration ... 61
Developer Account Registration ... 61

Apple Account Creation ... 61
Developer License Authorization ... 62
Xcode and iOS SDK Installation ... 63
Required Frameworks for Distribution ... 64
7.2 App Project Development ... 65
Creating Projects ... 65
XPLAFORM Library Setting ... 66
Additional Library/Framework Settings ... 67
Resource Settings ... 69
Image Settings ... 70
Message Settings ... 71
Build Environment Setting ... 73
AppDelegate.h ... 73
AppDelegate.m ... 74
main.m ... 74
Revise Additional Settings ... 75
Other Linker Flags ... 75
Header Search Paths ... 75
Build Active Architecture Only ... 76
Valid Architectures ... 76
Objective-C Automatic Reference Counting ... 76
sampleProject-Info.plist File Setting ... 76
Code Signing ... 77
7.3 App Test ... 79
8. App Development and Execution (Android) ... 80
8.1 App Development Environment Configuration ... 80
JDK(Java SE Development Kit) Installation ... 81
Download ... 81
Installation Check ... 83
Installing Eclipse ... 83
Installing Android SDK ... 84
ADT Plug-in Installation ... 86
8.2 App Project Development ... 89
Creating Projects ... 89
Nexacro Platform Library Settings ... 90
Resource Settings ... 91
Image Settings ... 92
Message Settings ... 93
Layout Settings ... 95
Build Environment Settings ... 96

MainActivity.java 96
AndroidManifest.xml 98
DeviceAPI 99
GlobalAPI 102
Asset/archive feature 102
Cache feature 102
8.3 App Test 103
8.4 Creating Installation Files 104
9. App Distribution 105
9.1 How to Distribute Archive Files 105
9.2 How to Distribute Apps through Android Markets 106
Google Play 106
SamsungApps 106
T Store 106
Olleh Market 107
U+ App Market 107
9.3 How to Distribute Android APK Files through Private Servers 107
9.4 How to Distribute iOS Apps 108

Part IV. Reference 111

10. Generating Code 112
10.1 Generating Code 112
10.2 Using nexacrogenerator.exe 113
Options 113
Examples 114
Generating Entire Application 114
Assigning Output Path User-specified Name Instead of XADL Name 114
Specifying Separate Library Path 115
Generating XADL File Independently 115
Generating Specific Service Path 116
Generating Specific Files 116
Generating Specific Theme 117
case 1) Theme registered in the cache folder 117
case 2) Theme registered in a relative path or Prefix path, based on an XADL file 117
case 3) Combination of case 1 and case 2 117
Saving Generation Result as Log File 118
10.3 nexacrocompressor.exe 118
Options 118
Examples 119

Compressing Specific JS Files ... 119
Compressing JS Files in Specific Directory ... 120
Specifying Suffix in Compressed File's Name. ... 120
Obfuscating Variables for Compressed File ... 120
Obfuscating Variables outside eval Function ... 121
11. Dataset XML Format ... 122
11.1 Dataset XML Layout ... 122
XML Declaration ... 123
XML Example ... 124
11.2 Dataset Elements ... 125
Root ... 125
Parameters ... 126
Parameters > Parameter ... 127
ID ... 127
Type ... 128
Dataset ... 129
Dataset > ColumnInfo ... 130
Dataset > ColumnInfo > ConstColumn ... 131
Dataset > ColumnInfo > Column ... 132
Dataset > Rows ... 133
Dataset > Rows > Row ... 134
Dataset > Rows > Row > Col ... 135
Dataset > Rows > Row > OrgRow ... 136
Dataset > Rows > Row > OrgRow > Col ... 137
Appendix A. Setting Path ... 138
A.1 Alias Path ... 138
A.2 nexacro.xml ... 139
A.3 Relative Path ... 140
Relative Path Support for Paths in Projects ... 140
Plugin Method ... 141
Appendix B. Exceptions ... 142
B.1 Web Browser Option ... 142
JavaScript Activation ... 142
File Download Activation ... 143
HTTP 1.1 Activation ... 143
XMLHTTP Activation ... 144
B.2 Internet Explorer Compatibility View ... 144
B.3 Adding Web Content Using Iframe ... 145
Appendix C. Installing Nexacro Platform X-APIs ... 147

C.1 Installation 147
C.2 License Authentication 147
C.3 Confirming Installation 148
Appendix D. Installing Nexacro-Xeni 150
D.1 Installation 150
D.2 Confirming Installation 152
D.3 Settings 153
D.4 Example 153
Export 154
Import 154

Preface

This manual covers administrative tasks and procedures to support the distribution and execution of applications developed using Nexacro Platform.

Target Audience

An administrator in charge of distribution or operation Nexacro Platform applications can refer to this manual for the operational procedures. Application developers can refer to this manual to understand how Nexacro Platform works and to run and test applications.

Related Documents

Nexacro Platform Programming

For details on Nexacro Platform application development, please refer to *Developer Guide*.

Nexacro Studio

For details on Nexacro Studio installation, composition and option, please refer to *Nexacro Studio Guide*.

Other Development Tools

For information about other development tools such as web servers not provided by TOBESOFT, refer to the manufacturer documentation.

Manual Contents

This manual is composed of the following items.

XPLAFORM Introduction

This chapter describes what Nexacro Platform is in general.

Nexacro Platform Outline

This chapter describes the basic structure and components of Nexacro Platform.

RUNTIME for iOS/Andriod

It describes packaging Nexacro Platform applications and distributing those applications to apps in the mobile environment.

Note

It describes additional contents needed in the process of distribution and management of applications.

Common Notation

To increase readers' understanding, this user manual contains separate notations for words or sentences with special meanings. The following is an explanation of the notations.

Notes, tips, and cautions are given as follows:

Notes are used to provide brief explanations, references, or comments.

Tips are provided to give helpful advice.

Cautions provide information that demands special attention from readers or users.

Part I.

Outline

1.

Nexacro Platform Introduction

An application with a high quality user interface and user experience (UI/UX) results in more rapid adoption and more productive users, leading to lower support costs, a reduced Total Cost of Ownership (TCO) and increased Return on Investment (ROI). The Nexacro Platform is a solution custom-tailored for business users, with all the features, functionality, and design elements required to support business requirements. The Nexacro Platform will empower you to create a high quality UI/UX, so that you can improve productivity and reduce repetitive data entry tasks in your business.

1.1 The Evolution of UI/UX for Business

The continued evolution of internet technology is now enabling connectivity for almost any physical object through the “Internet of Things.” As the scope of internet usage increases, web applications have also evolved to provide richer user experiences for corporate and individual users.

Prior technology generations included client-server (C/S) environments which provided acceptable performance, but included significant challenges associated with installation, distribution, and maintenance. Eventually, the World Wide Web fixed many of these issues, but slow data transfer speeds and limited UI functionality troubled users and developers. Several companies developed platforms to support the development of “Rich Internet Applications” to address these concerns while the HTML standard grew to enable more user interface technology over time.

Over time, the importance of usability has led application developers to focus not only on the user interface of an application but also on the full user experience with it. Those applications that deliver the best experiences to their users enjoy more rapid adoption and, since a good experience is an easy-to-use experience, these application lead to fewer support calls.

Recently enterprises have shown increasing interest in the "business view" where UI/UX is optimized for the business environment. Against this backdrop, we seek to improve the “Business User Experience,” including helping users make quick decisions and promoting productivity through an intuitive work environment, thereby creating business value.

BUX: Business User eXperience

Nexacro enables a great business-user experience by integrating three critical elements.

- A unified platform – One set of source code easily supporting different screen sizes, creating optimized results, across a wide variety of operating systems, browsers and devices, including tablets, phones and desktops.
- A unified development process - Nexacro Studio improves productivity by integrating all processes from development to testing and distribution while enabling simpler in-house development as well as advanced UI construction
- A data-centric unified data framework - capable of supporting structured and unstructured data as well as data from online transaction processing, online analysis processing, and big-data visualization applications.

1.2 Nexacro Platform

Nexacro Platform uses a JavaScript-based framework to accommodate enterprise demands. Applications developed using Nexacro Platform, with its write-once-deploy-many capability, support functionality on many different operating systems, browsers, and devices without requiring additional development.

Nexacro Studio, bundled with Nexacro Platform, is the WYSIWYG development environment that enables creation of mobile and web applications. Nexacro Studio offers developers a variety of technical capabilities including drag-and-drop editing of user interface components, an easy-to-use data binding capability, “Adaptive Visual Editor” - a feature that greatly simplifies support of varying device screen sizes and orientations, as well as a collection of interesting design effects, all of which collectively maximize development productivity.

The platform supports HTML5 and hybrid architectures using the same development process and setup files. HTML5 applications can be freely deployed without any special installation. Applications deployed with the in the hybrid architectures attain higher performance and reliability by leveraging an installed virtual engine optimized for each supported operating system.

Key Features

Table 1-1 Key Features

Feature	Description
Powerful IDE	Similar to existing development tools, Nexacro Studio, Nexacro Platform’s development environment, provides a variety of convenient development features. With its intelligent editor and WYSIWYG screen design, Nexacro Studio supports numerous convenient features for developers.
Support for standards	The Nexacro Platform uses widely adopted standards such as JavaScript, CSS, and XML. By using familiar technologies, developers don't need to learn new languages and can become productive in the new development environment quickly.
Integrated device event processing	By standardizing mouse, touch and other event processing, the same, single set of code can handle

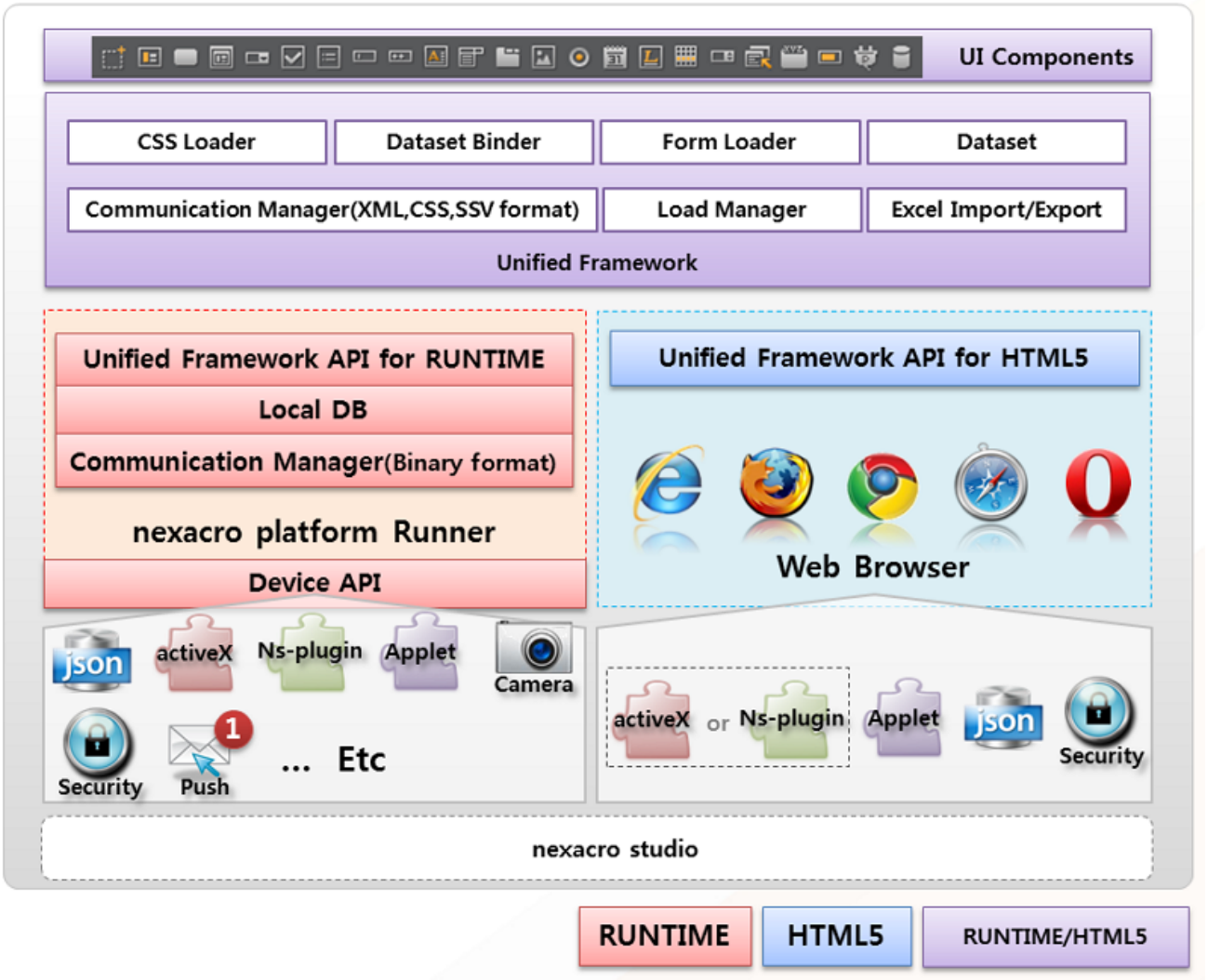
Feature	Description
	events on a variety of devices, reducing the cost of supporting multiple devices running multiple operating systems.
Quick data processing	Nexacro Platform enables responsive applications using optimized data communications and supporting asynchronous communications, which allows an optimal environment for each task.
Rich Component Library	Nexacro provides a variety of components for enterprise applications, reducing application development time. In particular, Grid, the most frequently used tool, provides strong support for externally-supplied data. With Nexacro Studio, you can create new events through drag-and-drop while a few simple clicks enable data binding for each component.
Hybrid Architecture	Although the performance of web applications is gradually improving, they still cannot always meet the performance requirements of the most intensive applications. Runtime provides a superior alternative.
Visual Effects	Its many styles and themes make it easy to create a variety of visual effects with Nexacro Studio.

2.
Nexacro Platform Outline

Nexacro Platform provides a unified framework with JavaScript-based components to support a wide range of devices and operating environments. With a single set of source code, Nexacro Platform can be used to develop applications for any environment.

2.1 Nexacro Platform Unified Framework

Nexacro Platform Unified Framework is the conceptual map of the Nexacro Platform, enabling cross-platform application development using a single code base. Under the Unified Framework, components are created with JavasScript, assuring identical behavior regardless of environment.



Component	Description
Unified Framework	Provides a basic framework for application configuration and development, enabling tasks including data communication/format (XML, CSV etc,) and inter-screen/inter-form communication. Also supports UI styles and themes.
UI Components	Collection of UI and dataset controls including: - UI Components: Button, Calendar, Checkbox, Grid, etc. - Basic Components: EditBase, FormBase, FrameBase, ScrollBar, etc. - Objects: Dataset
Unified Framework API	Enables applications to behave identically whether implemented as HTML5 or Hybrid applications. Developers can write code that is independent of the implementation architecture that will eventually be used. Functions supported by the API are implemented in C for Hybrid applications and JavaScript for HTML5. All code will access functions through JavaScript calls.
Device API	Enables access to various hardware capabilities that may be present on a device (e.g. camera, GPS, compass, phone, biometrics, accelerometer, etc.) - In general, browsers and devices prohibit the use of the Device API from HTML5 applications.

Component	Description
Local Database	Supports local data persistence when required by mobile applications. Supports SQLite and can be used by desktops and mobile applications. - Not supported for HTML5 applications
Binary Data Communication	Both Hybrid and HTML5 applications support communication using XML-, CSV- and SSV-formatted data. In addition, Hybrid applications can communication using binary data.

2.2 Nexacro Platform Components

Nexacro Platform is composed of four elements. The programming language and development environment are identical regardless of the application's implementation architecture; although, the distribution environment may differ.

Programming language

Nexacro Platform programming is divided into two parts - one to define user-visible screens visible and a second used for the scripts that implement business logic. Additionally, Nexacro Platform provides capabilities for applying and managing styles and themes in screens.

Form (screen) definition language (FDL) is XML-based and describes information such as each component’s properties, binding, and events. FDL is used to describe the multiple layouts created using the Adaptive Visual Editor. Lastly, FDL includes a separate file structure that documents environment information needed to run applications.

The following table identifies files created when developing applications using Nexacro Platform (note that these are all development-time files that are not necessarily included in the application at run-time):

Item	File Name (extension)	Purpose
Nexacro Platform project	*.xprj	• Project Information • TypeDefinition • Global variable • ADL (application definition language)
Nexacro Platform application definition	*.xadl	• Application operating environment • TypeDefinition • Global variable • Theme Information • Frame properties

Item	File Name (extension)	Purpose
		• Screen Info
Nexacro Platform form definition	*.xfdl	• Screen layout • Screen formproperties • Component properties • Additional layout • Script
TypeDefinition	default_typedef.xml	• Module • Component • Service • Update
GlobalVariable	globalvars.xml	• Global variable
Theme	*.xtheme	• Style Sheet (Frame, Form, Component etc.) • Image

As part of the build process, applications built with Nexacro Platform will be converted to JavaScript. At run-time, it is the JavaScript code that will run.

Development Environment

Nexacro Platform includes Nexacro Studio, Nexacro's integrated development environment (IDE), which includes a WYSIWYG interface. You can develop applications on Nexacro Studio without being concerned about the operating environment. Code created will be saved in Nexacro Platform formats and languages.

Although Nexacro Studio runs only on Microsoft Windows, applications developed using Studio may be run under any operating system or device with a web browser that supports HTML5.

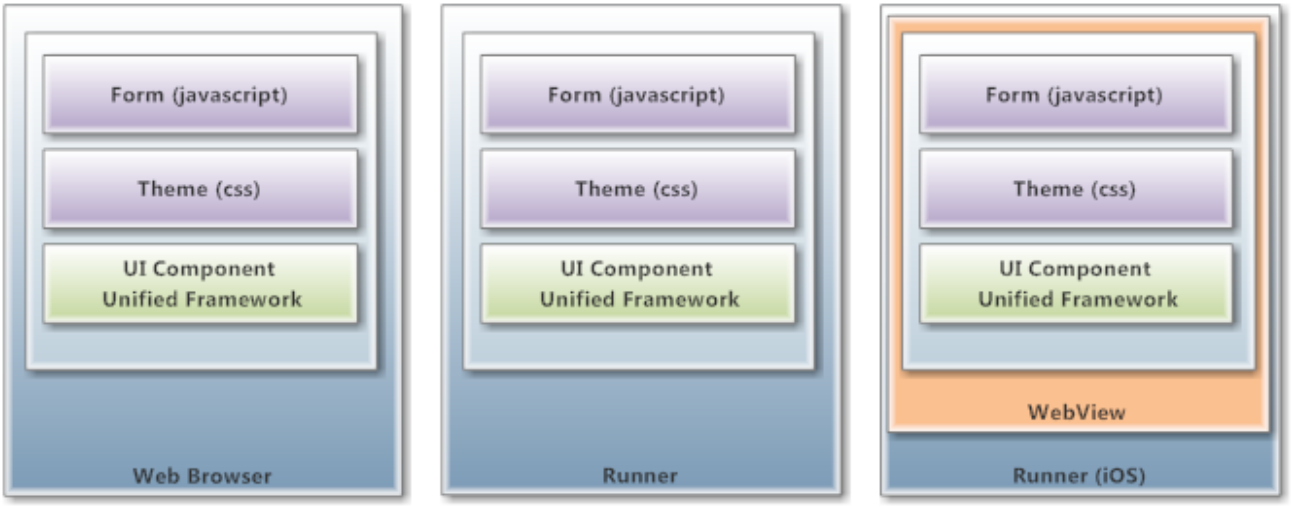
Depending on the run-time environment, you may need to perform additional work using different development tools andseparate authentication. For example, you may need an Xcode environment to distribute an iOS-based app.

Implementation Environment

Nexacro Platform runs applications through the Unified Framework and provides an optimized structure for each supported operating environment.

Nexacro Platform supports a Hybrid and an HTML5 run-time environment. The Hybrid environment offers an operating environment optimized for the operating system. As new operating systems or operating system versions are introduced, Nexacro Platform can readily be extended with APIs. The HTML5 version runs through standard capabilities provided by web browsers.

The architecture of the run-time environment is diagrammed below:



The Hybrid implementation includes Nexacro Platform Runner which implements an optimized rendering engine. For this reason, hybrid applications deliver faster, more reliable performance than HTML5 applications.

Even in hybrid applications, operating systems may impose restrictions.. For example, an app running on iOS must use an object called UIWebView. Although the app makes use of the same framework, under the operating system manufacturer's policy, it must be encapsulated with the UIWebView object.

An HTML5 application's performance may vary based upon the web browser used. For example, Microsoft ended its technical support for Internet Explorer 6 in April 2014, after which there is no guarantee of performance and no effective way to reduce security risks.

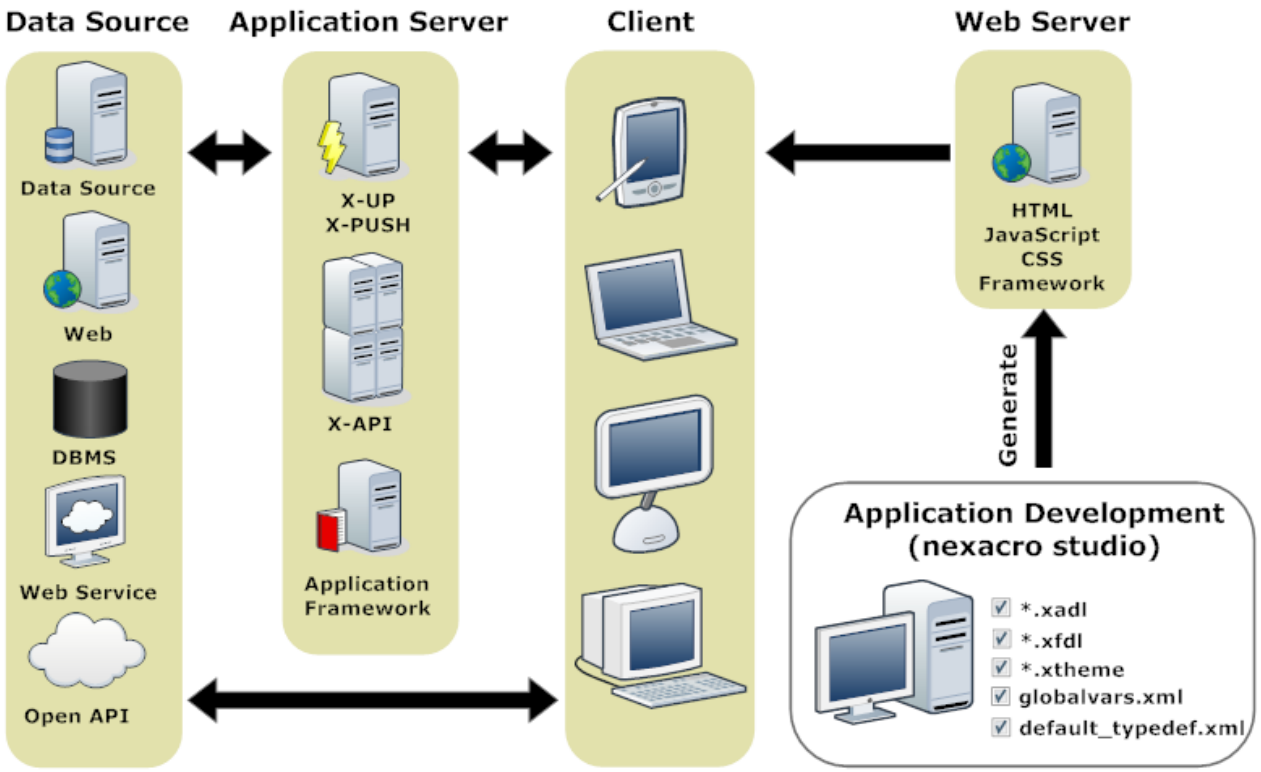
The Run-time Environment

Distribution refers to the tasks required to install an application and the resources on which it depends. Depending on the user's environment and application, you may need to download applications and modules developed using Nexacro Platform and install them on the user's device (whether mobile, laptop or desktop).

Nexacro Platform applications run on client machines that the user will work on. The application may need to communicate with the Nexacro Application Server for tasks such as data processing, which can be performed by using the X-API module. Where real-time data must be processed on the client machine, an additional module, X-Push, may be used.

For basic distribution tasks, HTTP is the simplest protocol to use. In environments where Internet access is unavailable, you may need a separate medium to distribute resources and applications.

Additional steps may be required depending upon the run-time environment of the application. For further details on distribution and other issues, please refer to the Nexacro Administrator Guide.



X-PUSH is compatible only with hybrid applications.
The installation modules are distributed as part of the execution environments for hybrid applications.

When you use a theme provided by Nexacro Studio, a theme file is converted from the basic theme.

2.3 System requirements

The environments in which Nexacro Platform applications can work are as follows:

- Runtime

Item	Specs	Windows	Android	iOS
CPU	Minimum	Intel Celeron Processor 430	ARM Cortex-A8 600 MHz	ARM Cortex-A8 600 MHz
	Recommended	Intel Pentium 4 1GHz	ARM Cortex-A8 Over 1 GHz	ARM Cortex-A8 Over 1 GHz
Display	Minimum	-	HVGA	HVGA
	Recommended	-	HVGA or higher	HVGA or higher
Memory	Minimum	512M	256M	256M
	Recommended	1G	512M	512M
HDD(ROM)	Minimum	100M	4M	4M
	Recommended	200M	10M	10M
Platform	Minimum	Windows2000 SP4	Android 2.3	iOS 4
	Recommended	Windows XP SP2	Android 2.3	iOS 6

Devices capable of supporting Nexacro Platform are as follows:
The following devices have been verified by us and are supported without special request from customers

• Android Smart Phone
Samsung Electronics
- Galaxy S3 (Android 4.1.2)
- Galaxy S4 (Android 4.2.2)
- Galaxy Note 8.0 (Android 4.1.2)
- Galaxy Note 10.1 (Android 4.1.1)

LG Electronics
- Optimus G2 (Andriod 4.2.2)
- Optimus Vu2 (Andriod 4.0.4)

SHARP
- Aquos Phone (Andriod 4.0.4)

• Android Tablet
Samsung Electronics
- Galaxy Tab 10.1 (Android 3.2)

Fujitsu Mobile

- Arrows Tab Wi-Fi FAR70A (Android 3.2)

• iOS

iPhone5 (iOS 6.0.1)

iPhone5s (iOS 7.1)

iPad2 (iOS 7.1.1)

iPad4 (iOS 7.1.1)

iPad mini (iOS 6.0.1, 7.1)

- HTML5

Web browser	Minimum specs	Recommended specs
Google Chrome	10.0.648.127	22.0.1229.79
Firefox	8.0.1	14.0.1
Internet Explorer	8.0	10.0.9200.16521
Opera	11.0	12
Safari	4	6.0.2

Mobile Web browser can only support the following browsers .

• Android default browser

• Android Google chrome browser

• iOS default browser

Mobile Web browser	Minimum specs	Recommended specs
Android default browser	Android 2.3 default browser	Android 2.3 default browser
Android Google chrome browser	Android Chrome (Android 4.0)	Android Chrome (Android 4.4)
iOS default browser	iOS 4 default browser	iOS 6.0.1 default browser

3.

Nexacro Platform Distribution

To use applications developed with the Nexacro Platform, an appropriate environment must be created on the user's machine. Certain files must be downloaded to the user environment. The process of establishing this environment and downloading files is called "Distribution." Distributed files are managed automatically based upon the cache policy and are replaced with new files as necessary.

For the Nexacro Platform HTML5, applications are executed in the currently used browser environment. For this reason, applications are executed directly by downloading the necessary files from a web server.

For the Nexacro Platform Run-time, additional applications must be installed based on the execution environment that will be used. Please refer to information in this guide about the corresponding Runtime version distribution.

3.1 Distribution Overview

To run Nexacro Platform applications, you need several files. The distribution files for each execution environment are listed in the following table.

Item	Description	windows	iOS/andriod	HTML5
Runtime Engine	Script, Memory, Rendering Process	O	□ (iOS: WebView)	X (Web browser)
plugin	Plug-in for each browser	O	X	X
Fremework	Integrated framework (Java Script)	O (XFrameworkLib.dll)	O	O
Component	Component Library (Java Script)	O	O	O
Application	Application Code (Java Script)	O	O	O

The big difference between the Runtime version and the HTML5 version is the Runtime engine. HTML5 version directly uses frameworks, components and applications needed for a web browser. On the other hand,

the Runtime version needs to separately distribute the Runtime engine appropriate to each execution environment.

Although it might seem a burden for the user to distribute the Runtime engine, the Runtime version can be used to make a connection to the existing system according to the system environment, or to use expanded functions which cannot be provided on the web browser.

The basic functions provided by Nexacro Platform for both versions are same. However, only the Runtime version supports expanded functions such as device APIs or the interconnection to external devices.

3.2 Distribution Files

This section describes each distribution in detail.

Please refer to A.1 Alias Pathfor alias paths used by the Nexacro Platform to set paths on the user's computer.

The of name of files and libraries listed herein may change depending on distribution updates.

Runtime for Windows

After distributing the Nexacro Platform Runtime engine, call applications according to their execution methods. The ontents are the same for the HTML5 version. However, with the HTML5 version, frameworks and JavaScript files are not be downloaded from the server, and instead are included in the dll format of library files and then distributed.

Item	File	Description	Location
Runtime Engine	v8.dll	V8 Script Engine Library	%nexacro%
	XMemLib.dll	Memory Management Library	%nexacro%
	XBasicLib.dll	Base Libaray	%nexacro%
	nexacrolib.dll	Platform & Render Engine Library	%nexacro%
	nexacro.exe	Nexacro Platform Executor	%nexacro%
ActiveX DLL	nexacrolauncherax14.dll	Nexacro Platform Launcher	%SYSTEM32%

Item	File	Description	Location
	nexacroax14.dll	Plug-in for each Browser	%nexacro%
Plugins DLL	npnexacrolauncher14.dll	Nexacro Platform Launcher	%SYSTEM32%
	npnexacro14.dll	Plug-in for each Browser	%nexacro%
Framework DLL	XFrameworkLib.dll	Framework API Library	%nexacro%
	XMobileApi.dll	Device API Library	%nexacro%
Extend DLL Files	protocol adaptor dll	Network Protocol Adaptor	%UPDATE%
	External DLL	External DLLs	%UPDATE%
Runtime Cache	cache.db	Runtime Cache DB	%CACHE%
	cachedfiles	Runtime Cached Data Files	%CACHE%\key_adl
Runtime Config Files	nexacro.xml	Runtime Execution Environment File & UserProfile	%USERAPP%
Log Files	nexacro_xxxxx.log	Log File	%USERAPP%\Log
Launcher Download Files	splash.png	Splash Image File	%USERAPP%
	loadingimage.png	WaitCursor Image File	%USERAPP%
	nexacro.ico	Application Icon File	%USERAPP%
	globalvars.dat	Global Variable List File	%USERAPP%

HTML5

This distribution directly loads all of the required files on the web browsers. It performs differently depending on web browser in use.

Item	File	Description	Location
Execute HTML	index.html		%WEBDEPLOY PROJECT%
Framework JS	BasicObjs.js		%WEBDEPLOY FRAMEWORK%
	SystemBase.js	System Utility	%WEBDEPLOY FRAMEWORK%
	Platform.js	Platform Objects	%WEBDEPLOY FRAMEWORK%
	CssObjs.js	Style Object	%WEBDEPLOY FRAMEWORK%
	ErrorDefine.js	Error Information	%WEBDEPLOY FRAMEWORK%
Component JS	CompBase.json	Component Base Module	%WEBDEPLOY COMPONENT%
	ComComp.json	Common Component Module	%WEBDEPLOY COMPONENT%
	Grid.json	Grid Component Module	%WEBDEPLOY COMPONENT%
	DeviceAPI.json	DeviceAPI Module	%WEBDEPLOY COMPONENT%
Resource Files	waitimage.gif	WaitCursor Image File	%WEBDEPLOY RESOURCE%
Theme File	defaulttheme.js	Deployed Theme Source	%WEBDEPLOY THEME%
Application Files		Deployed Application Source	%WEBDEPLOY PROJECT%

RUNTIME for iOS/Android

iOS/Android applications need separate packaging processes for the different operation systems. It generates application sources created on Nexacro Studio as archive files and creates distribution files appropriate to each operating system.

Item	File	Description	Packing
Android App Files	nexacro14.jar	Android Library Jar	Android App
	libnexacro14.so	Android Library SO	
	startinfo_android.json	Android Archive Information	
	strings.xml	Android Locale String Information	
iOS App Files	nexacro14.framework	iOS Library Framework	iOS App
	startinfo_ios.json	iOS Archive Information	
	localizable.strings	iOS Locale String Information	
Execute HTML	hybrid.html		Hybrid Run Archive
Framework JS			Hybrid Engine Archive
Component JS			Hybrid Engine Archive
Resource Files	loadingimage.png		Hybrid Engine Archive
Theme File	defaulttheme.js		Hybrid Theme Archive
Application Files			Hybrid Application Archive

Part II.

Runtime for Windows

4.
Outline of Nexacro Platform Distribution

The Nexacro Platform execution environment must be installed on users' Windows PCs prior to downloading and running applications that use it.

The Nexacro Platform execution environment can be downloaded through HTTP or offline, using a manual procedure.

4.1 Phases of Distribution

Distribution of the Runtime version covers up to a call for the first HTML file after the engine distribution. Nexacro Platform application users should abide by the process order for HTML files and JavaScript files described herein.

1. Nexacro Platform Engine Distribution
   This installs the Nexacro Platform engine, which manages all stages of distribution.
   Install the engine by downloading it from the web or by receiving an execution file ".exe". If you download the file from a web browser, use the CAB file provided by TOBESOFT.

2. Integrated Framework and Component Library Distribution
   A framework will be distributed as a resource file to the Framework DLL file distributed with the engine.

3. Application Distribution
   Applications developed by developers will be distributed. Component modules are distributed as well.
   Application distribution uses networks efficiently through the "Intelligent Cache" technique.

4.2 Distribution Flow

Most of the distribution process is performed by the Nexacro Platform Engine. The Nexacro Platform Engine is usually installed on the user device as a CAB file through a web browser. However, if the CAB process fails, separate exe files are provided.

Phase 1 : Nexacro Platform Engine Setup

From the Web browser, the Nexacro Platform Engine is installed using the CAB file (n *exacro14_SetupEngine.cab*). However, this method requires the insertion of the following coding.

```
<HTML>
<HEAD>
<TITLE> Plugin sample page </TITLE>
<BODY>
<object id="NexacroAXCtrl"
    classid="clsid:6DB5422D-536F-4B80-B32C-16BEA0971512"
    CodeBase="./nexacro14_SetupEngine.cab#VERSION=2014,7,11,2">
</object>
</BODY>
</HTML>
```

The version number corresponds to the module version being set up. Refer to the *VersionInfo* file distributed with the product.

Phase 2 : Setting Nexacro Platform Application Program Execute Environment

To run an application program, you need the ADL file which has the environment information of application program. This is because the TXPLATOFRM Engine downloads and analyzes the ADL file for application program distribution and composition.

The ADL (Application Definition Language) file defines the elements needed to configure application programs. It is saved with the xadl extension and includes the information about the TypeDefinition and GlobalVariable as a JavaScript file when distributed.

The ADL also defines other elements such as the basic screen arrangement, component and object composition and theme composition.

Phase 3 : Application Program Execution

After configuring the application program according to the ADL file, the Nexacro Platform Engine downloads and outputs the UI screen.

An FDL file (Form Definition Language) with an xfdl extension is saved and provided as a JavaScript file during distribution. It contains the UI information and the logic information to be processed within a screen.

4.3 Setup Files

Various files needed for the Nexacro Platform distribution are also included in the setup files, such as modules, licenses and functions.

Modules

The following are the main modules included in the XPLATOFRM setup file.

Module Setup File	Modules
nexacro14_SetupDeveloper.exe	• Nexacro Platform Engine (nexacro.exe + library dll) • nexacro14lib (JavaScript) • Nexacro Studio
nexacro14_SetupEngine.exe nexacro14_SetupEngine.cab	• Nexacro Platform Engine • nexacroax14.dll • nexacrolauncherax14.dll
nexacro14_Launcher.cab	• nexacrolauncherax14.dll

License files

Following are the types of license files.

License file	Description
nexacro14_server_license.xml	Provides file in XML format In the same place as X-API jar file or in classes folder Checked when X-API, HTML5 is running
nexacro14_client_license.xml	Provided as a file in XML format Placed in route approachable by web Sets relevant file route and file name in license URL of ADL Checked when runtime and widget is running
Tool License	Provided as a file in Product Key format Help of Nexacro Studio – Registered in about Nexacro Studio.

Functions of Main Modules

The following are the main functions of the distributed modules.

Module Name	Main Function
nexacrolauncherax	Downloads icon image Creates shortcuts Drives Nexacro Platform Engine
nexacroax	Drives Nexacro Platform Engine on web browser

4.4 Types of Nexacro Platform Distributions

The distribution of the Nexacro Platform differs according to in what form Nexacro Platform Runtime is executed.

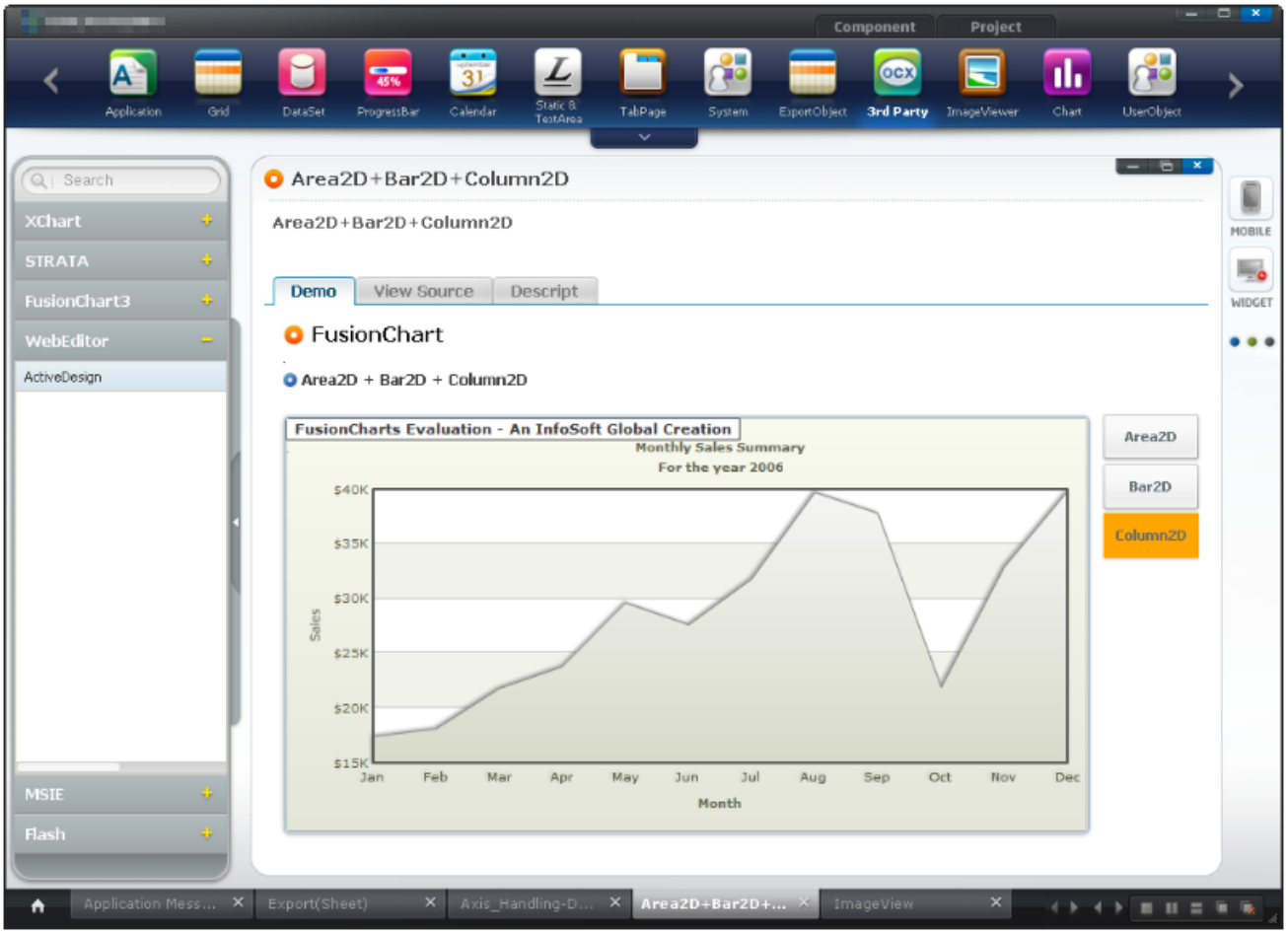
Runtime Execution Form

The Runtime Execution is separated into two forms according to how the core part of Nexacro Platform Runtime, the engine, is run. These are “Execution Version Execution” and “Web Browser Plugin Version Execution”.

These execution forms are further differentiated based on how the distribution is performed.

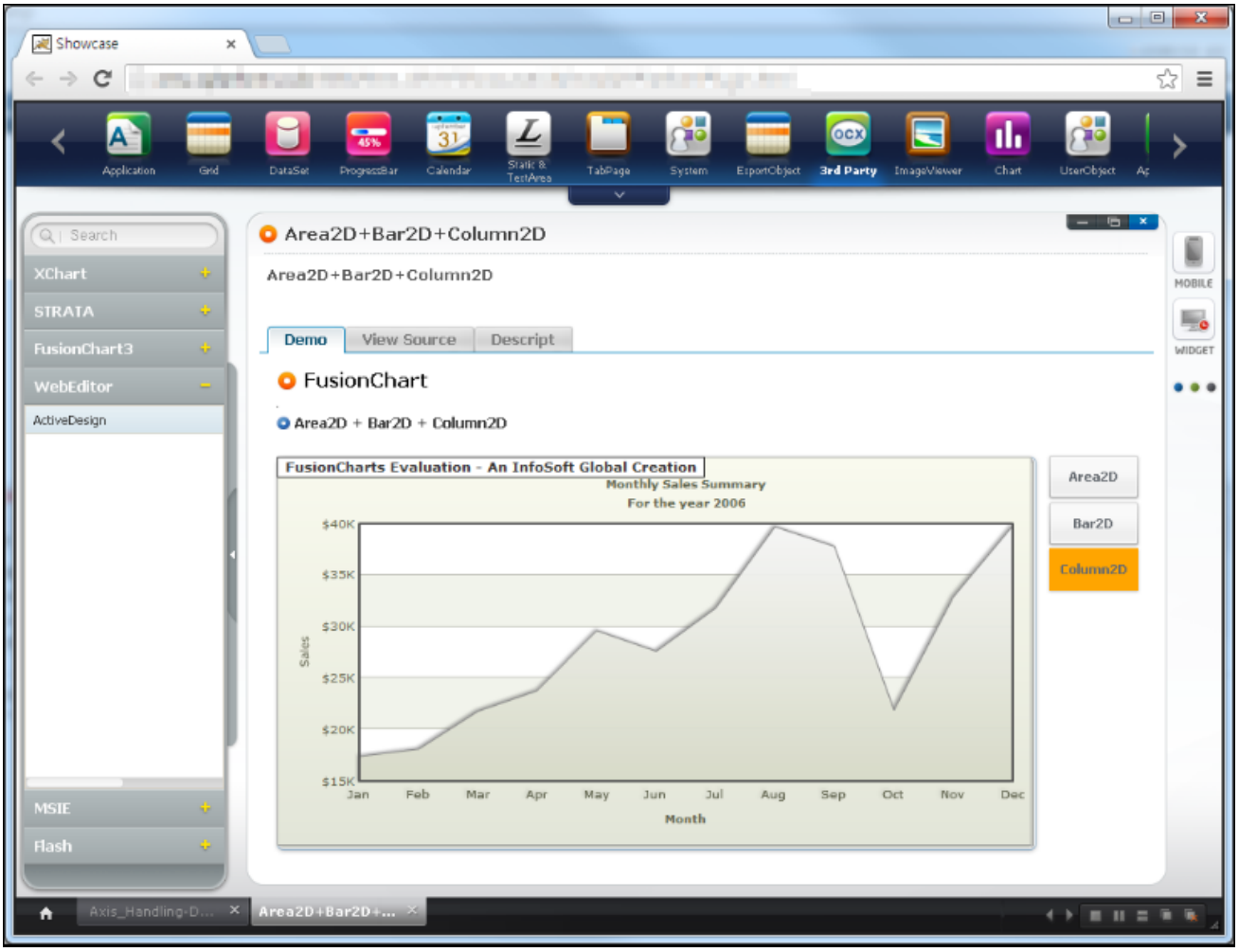
Execution Version Execution

The Nexacro Platform is run as another window by launcher and exe file.



Web Browser Plug-in Version Execution

The Nexacro Platform runtime is run as an embedded form on the web browser.



Distribution by Web Browser Plug-in is processed independently. This could provide unpredictable results.
If you want to use a web browser for the execution, consider using Nexacro Platform HTML5 version.

Types of Basic Distribution

Distribution of Nexacro Platform can be divided into three distribution types, depending on the Nexacro Platform execution form:

- Plugin distribution
- Launcher distribution
- exe distribution

These are compared in the following chart.

Type Category	Web Browser Plugin Distribution	Launcher Distribution	exe Distribution
Running window	Web Browser	Nexacro Platform Runtime	Nexacro Platform Runtime
Setup file	nexacro14_SetupEngine.cab	nexacro14_SetupEngine.cab	nexacro14_SetupEngine.exe
		nexacro14_SetupEngine.exe nexacro14_Launcher.cab	
Engine execution	Nexacro Platform ActiveX execution	nexacrolauncherax execution	nexacro.exe execution
Usage of shortcut	None	Create shortcut	Use shortcut

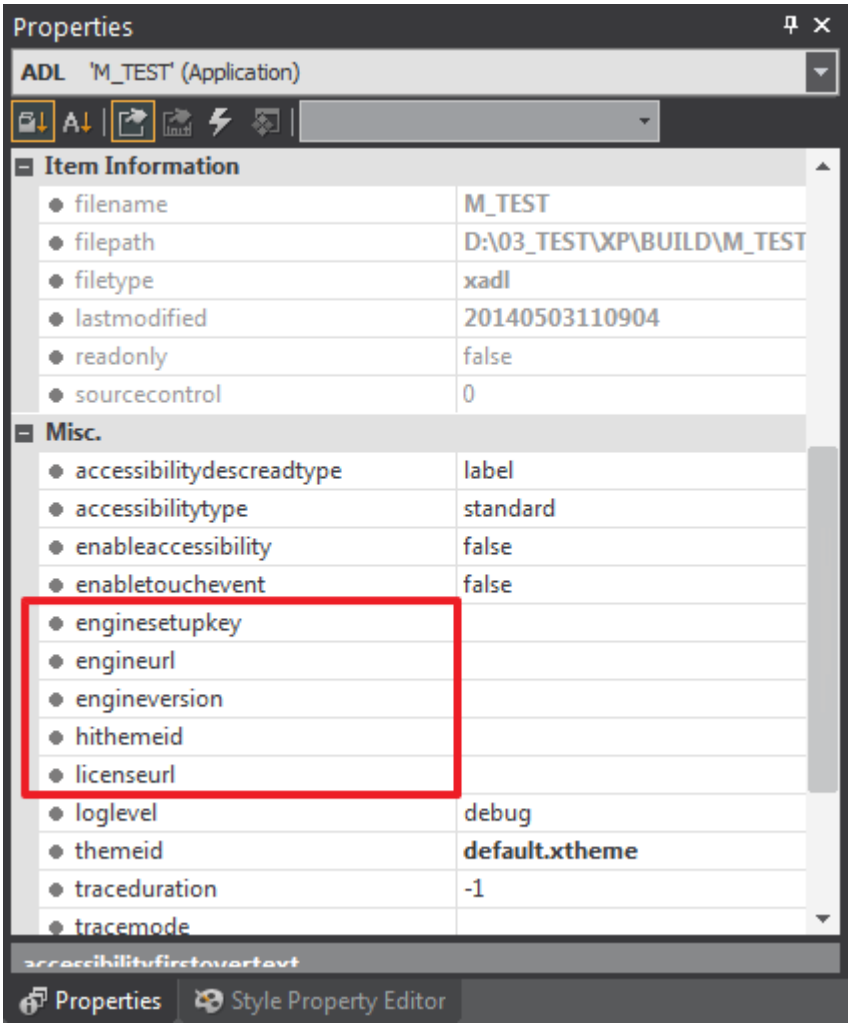
To use shortcuts, create the shortcuts using Launcher, and execute them by clicking on the corresponding icon using exe execution.

4.5 Version Administration

Most distributions are directed by the Nexacro Platform Engine. However, because the engine is usually distributed through web browser, how are upgrades distributed? The Nexacro Platform engine upgrades by itself.

Engine Upgrade Information Registration

The ADL files have the upgrade information about the engine, as illustrated in the following figure of the ADL property screen in Nexacro Studio.



The following table lists the property details.

Property Name	Description
enginesetupkey	Product key of the setup file

Property Name	Description
	When using Nexacro Platform Engine, you do not need to enter any value. However, when the developer is writing a separate setup file, the product key of the particular setup file is entered here.
engineurl	Path to the server URL with an existing setup file
engineversion	Version of the setup file
licenseurl	Designates the path to the license file

The following HTML code shows the relationship between the value used in the HTML file when the engine was installed and the ADL property value.

```
<HTML>
<HEAD>
<TITLE>Plugin sample page </TITLE>
<BODY>
     <object  id="NexacroAXCtrl"
          classid="clsid:6DB5422D-536F-4B80-B32C-16BEA0971512"
          CodeBase="./nexacro14_SetupEngine.cab# ? engineurl
          VERSION="2014,7,11,2"> ? engineversion
     </object>
</BODY>
</HTML>
```

Executing Nexacro Platform Engine Upgrade

After inputting the ADL property value with Nexacro Studio and uploading the ADL file to the distribution server, the Nexacro Platform Engine will automatically upgrade after comparing the engine URL and engine version value in this ADL file with the Nexacro Platform engine that is already installed.

The version of the already installed engine can be checked in **Settings/Add or remove program**.

When upgrading Nexacro Platform engine, the Nexacro Platform Engine will be rerun.

Part III.

Runtime for iOS/Android

5.

App Development Outline

With Nexacro Platform, you can develop applications to support a variety of operating systems and user environments using a single set of code. However, when distributing applications, some functions and distribution can vary depending on the user environments.

This chapter covers information developers and system administrators will need for the iOS and Android operating systems for smartphones and tablet PCs.

5.1 Technology Components

Applications distributed to iOS and Android operating systems can run even when devices are in motion. They can utilize a variety of equipped sensor functions. In the case of smartphones, the applications can be connected to functions such as call or SMS. Therefore, it is necessary for you to understand device APIs, development, execution and distribution.

Device APIs

The device APIs provided on the Nexacro Platform are listed in the following table.

Device API	Contents of Function
System	Basic terminal information Making calls Playing simple-sound files Objects > Misc. Objects > System
Acceleration	Terminal acceleration information by using the accelerometer sensor
LiteDB	Local DB usage
VirtualFile	File input/output
AudioPlayer	Sound file play control
AudioRecorder	Audio recording

Device API	Contents of Function
Camera	Built-in camera
ContactSet	Address book inquiry/add/delete/edit
ExternalAPI	Access/connection to external devices
GeoLocation	Terminal location information by using GPS and WPS: ● GPS - Global Positioning System ● WPS - WiFi Positioning System
ImagePicker	Terminal album access
Map	Map component for map marking
Network	Terminal network status check (No Network, 3G, Wi-Fi) Available communication check with certain sites
Sms	SMS transmission SMS reception detection Received SMS storage check/delete
Vibrator	Vibration generation

Provided functions can be added or changed depending on versions. Which functions are supported depends on the device operating system. Please refer to the reference guide included in Nexacro Studio for details.

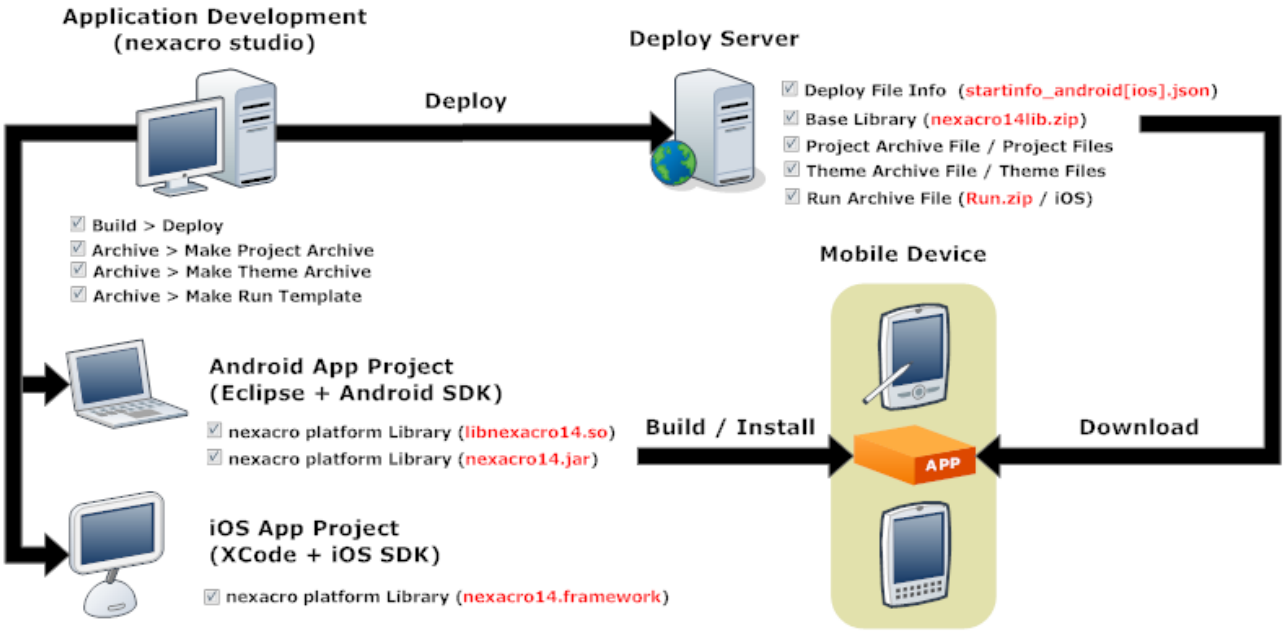
Development

The way to develop screens and services is the same in both versions. The difference is that device APIs which are used in the iOS and Android operating systems can be utilized as additional components. The basic test for newly created applications can be done on Nexacro Studio, but the actual operation of device APIs should be tested by distributing the applications to the devices.

In both iOS and Android, unverified applications cannot be executed, and only app-type applications can be executed. For these reasons, applications developed on Nexacro Platform should be developed and distributed as apps so that they can be directly executed on devices, not on web browsers.

Nexacro Platform provides libraries needed for Android and iOS development environments to make it easy to develop apps. Applications based on packaging (archive file creation) can also be distributed.

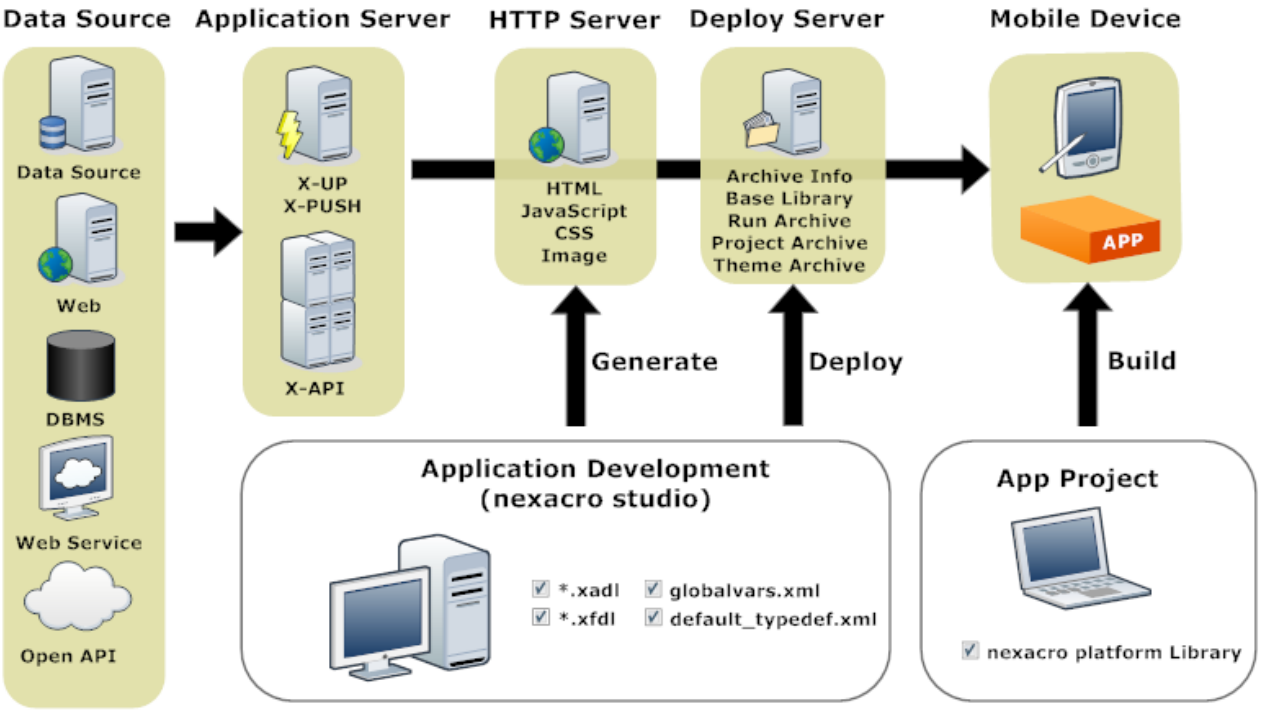
Nexacro Platform-based apps operating in iOS and Android are developed using the following process.



Nexacro Studio does not support developing apps that can be directly distributed to markets or tested on simulators.

Execution

An application developed on the Nexacro Platform can be installed and run by downloading the required file specified in the *startinfo_android[ios].json* file in the designated deploy server, or by updating a previously modified file.



The web server and the deployment server do not need to be physically separated. You can adjust equipment deployment, if necessary, as long as you designate a path for each server.

Distribution Environment

Distribution reflects creating archive files to a deployment server and installing Nexacro Platform applications on devices.

Apps are distributed through markets supported by each device's operating system.

For example, for Android, apps are distributed through various markets run by Google, device manufacturers and telecommunications service providers. Apps can be distributed through servers.

In the case of iOS, the APP Store is used for distributing apps. For apps to be distributed only for the internal use at companies, the Enterprise Program among iOS developer programs can be utilized.

Apps can be distributed on Android and iOS as follows.

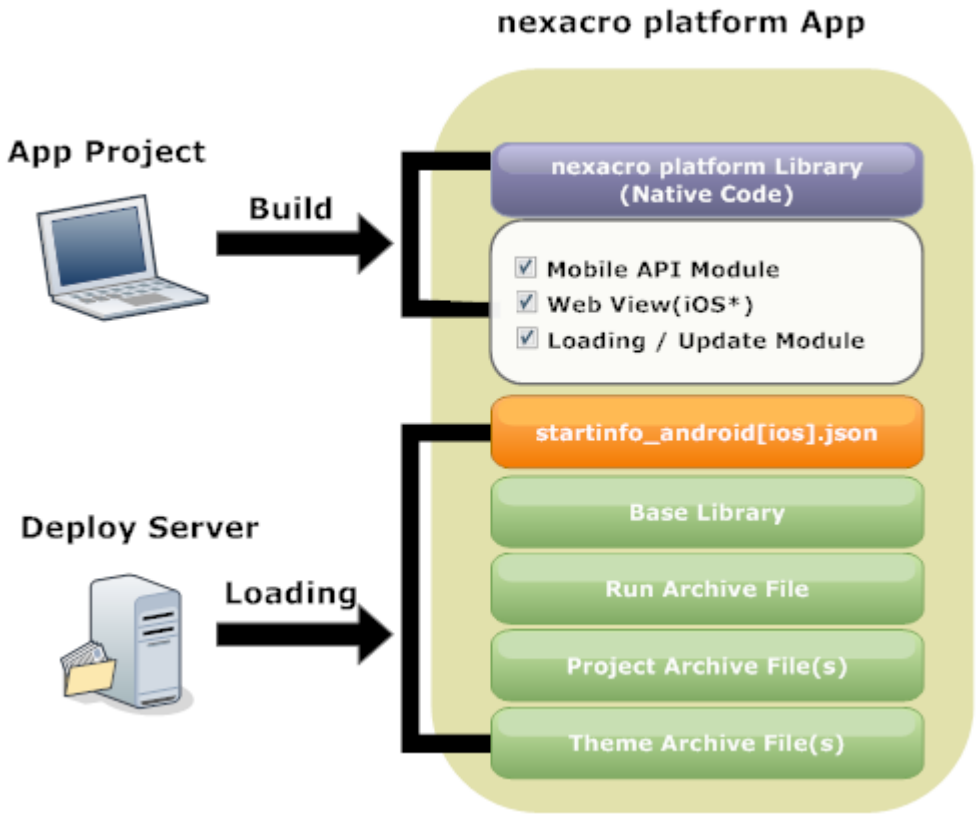
OS	Method	Market	Description	Operator
Android	Market	Google Play	Run by Google which developed Android. The largest number of apps are distributed in the market.	Google
		Samsung Apps	Run by Samsung Electronics, a device manufacturer	Samsung Electronics
		T store	Run by SK Telecom, a communications service provider	SK Telecom
		Olleh Market	Run by KT, a communications service provider	KT
		U+ App Market	Run by LG U+, a communications service provider	LG U+
	Self-distribution		Apk files are located in private web servers and distributed through the corresponding URLs	Self-operation
iOS	Market	Apple Appstore	Run by Apple Private servers are not available	Apple
	Self-distribution		Available by joining the Enterprise Program https://developer.apple.com/programs/ios/enterprise	Self-operation

Available markets for the Android operating system can be different from countries providing the service. The table above specifies the major markets available in Korea.

5.2 Development Process

Major Components

Nexacro Platform apps are mainly composed of the following elements.



The following table describes the main components of Nexacro Platform apps.

Component	Detail	Time of Inclusion
nexacro platform Library (Native Code)	Included when an app is developed by the library. Provided as: ● iOS(nexacro14.framework) ● Android(nexacro14.jar, libnexacro14.so) The composition of the library is as follows: ● Mobile API module Native code for Mobile API operation ● WebView (iOS) Web View control for displays such as screens ● Loading/updating module A module for loading apps or updating apps	Build includes Nexacro Platform application projects on Eclipse or XCode

Component	Detail	Time of Inclusion
	Note ● nexacro14.framework which is used on the iOS operating system is provided as a compressed file that is unzipped and used as a folder. The files used for testing on a device verses on a simulator are different.	
startinfo_android.json startinfo_ios.json	Includes path information of the following files and reads the information by accessing the designated path when building an app: ● Run Archive File (iOS) ● Project Archive File ● Theme Archive File ● Nexacro Platform Library **Note** ● The File name can be changed, but it shall be same with the values specified on Eclipse or XCode.	Downloaded into Nexacro Platform apps by loading or updating it at the time of execution.
Base Library	It provides Nexacro Platform frameworks by compressing them. It uses the nexacro14lib.zip file which includes components realized by Java Script. **Note** ● The File name can be changed, but it shall be same with the value specified on Eclipse or XCode. ● The file should be made by compressing the nexacro14lib folder structure provided by TOBESOFT.	
Run Archive File	It contains the information about components to be used on Nexacro Platform and a path of an ADL Java Script file to be executed for the first time. **Note** ● Required only when using iOS operating system ● Provided file names (Run.zip, Run.html) are set internally on Nexacro Platform. Do not change the names. Otherwise, they will not work.	
Project Archive File / Project File	A Nexacro Platform HTML5 project file developed on Nexacro Studio **Note** ● It is recommended to distribute a project archive file from the deploy server. The archive file shall be distributed as the form of Archive00.xzip. ● Depending on the settings, it is possible to access a project on the HTTP server without distributing the archive file. The operating mode can be selected based on the distributed app's characteristic.	
Theme Archive File / Theme File	It is a theme file of a Nexacro Platform HTML5 project. **Note**	

Component	Detail	Time of Inclusion
	● Provided file name ([Themename].zip) is set internally on Nexacro Platform. Do not change the name; otherwise, it will not work. ● Depending on the settings, it is possible to access a project on the HTTP server without distributing the archive file. The operating mode can be selected depending on the distributed app's characteristics.	

Development and Execution Process

The development and execution process is listed in the following table.

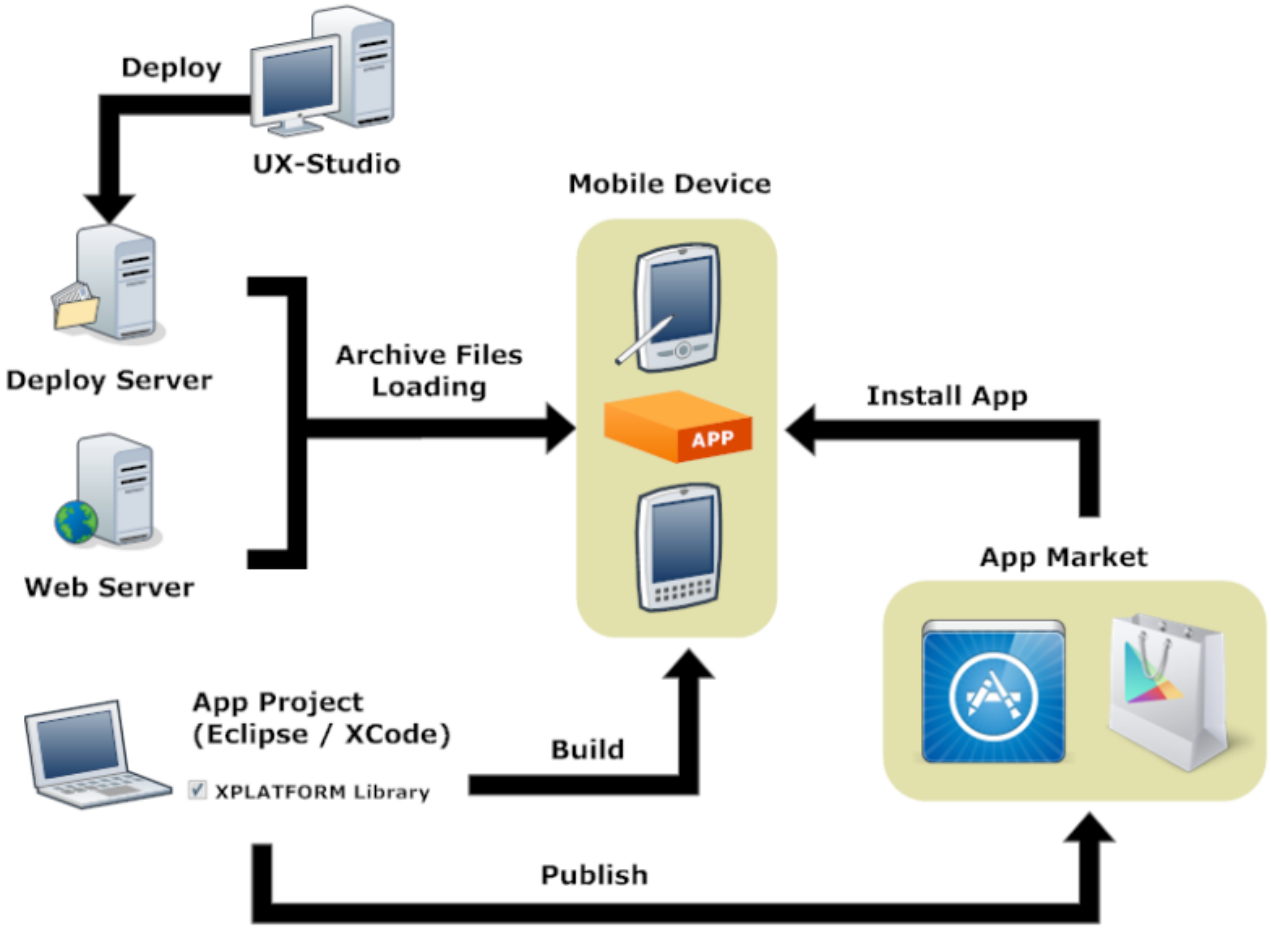
Phase	Detail
Project Development	Nexacro Platform applications are developed on Nexacro Studio.
Deploy Server Reflection	Required archive files are created and distributed on Nexacro Studio and then reflected to the server.
App Project Creation / Build	Application projects are created on Eclipse(Android) and XCode(iOS). The provided Nexacro Platform Library is referred after the projects are created. After configuring the project settings, apps (APK/APP) are built on Eclipse(Android) and XCode(iOS). □ For iOS, an Apple developer account is needed to build the apps for the devices.
Execution	After connection to devices, apps are executed from Eclipse(Android) and XCode(iOS) to the devices. □ For iOS, an Apple developer account is needed to execute the apps on the devices.

Distribution Process and Method

Archive files and apps are distributed separately on Nexacro Platform as illustrated in the following graphic.

Archive files, composed of base libraries, project archives, theme archives, run archives and etc, are packaged on Nexacro Studio and delivered through the deploy server.

Apps are created in the created projects, including Nexacro Platform libraries, and are distributed to markets or private servers (only in case of Android) as files available to be distributed. When a user runs a smartphone and executes an app that has been downloaded from a market or a private server, the app loads the required information from the deploy server and displays the app on the screen.



Nexacro Platform hybrid apps (APK/APP) are distributed through markets as follows (details vary depending on markets - refer to the manuals related to each market):

1. Developer account creation
2. App creation for distribution
3. Inspection registration
4. Inspection completion
5. Service start

5.3 Specifications for App Development and Execution

This section lists specifications for devices and SDK versions for Android and iOS.

Use the latest version of tools and SDK for distribution.

Android

Section	Item	Description
Development	OS	Window/Mac/Linux are available
	Eclipse	An integrated programming development environment to support various languages such as Java. It is used to develop Android apps (free download) http://www.eclipse.org/downloads/
	ADT	An Eclipse plug-in provided by Google to develop Android apps (free download) http://developer.android.com/tools/sdk/eclipse-adt.html
	Android SDK	An Android SDK used on Eclipse equipped with ADT (free download) Depending on the Android SDK version, certain specifications for OS and Eclipse might be required Minimum : Android 3.0 SDK or later http://developer.android.com/sdk/index.html
	Developer Account	A developer account for each company is needed to distribute apps to Google Play/Samsung Apps/T store/Olleh Market/U+ App Market. The costs are different depending on companies.
Device	Device OS	Android OS 2.3 or later □ **However, other operating systems will be examined.**

iOS

Section	Item	Description
Development	OS	Apps can be developed/distributed only on Mac OS X Minimum : Mac 10.6.8 or later
	XCode	Tool to develop apps for iOS (free download) https://developer.apple.com/xcode/

Section	Item	Description
	iOS SDK	Installed when XCode is installed (free download) Apps can be distributed to Appstore only when they are built with the latest version of SDK https://developer.apple.com/technologies/ios6/
	Developer Account	When created on Apple iOS developer page, the cost will be charged annually (approximatly $99/year) https://developer.apple.com/programs/ios/
Device	Device OS	iOS 4 or later □ **However, other operating systems will be examined**

Devices capable of supporting Nexacro Platform are as follows:
The following devices have been verified by us and are supported without special request from customers

• Android Smart Phone
Samsung Electronics
- Galaxy S3 (Android 4.1.2)
- Galaxy S4 (Android 4.2.2)
- Galaxy Note 8.0 (Android 4.1.2)
- Galaxy Note 10.1 (Android 4.1.1)

LG Electronics
- Optimus G2 (Andriod 4.2.2)
- Optimus Vu2 (Andriod 4.0.4)

SHARP
- Aquos Phone (Andriod 4.0.4)

• Android Tablet
Samsung Electronics
- Galaxy Tab 10.1 (Android 3.2)

Fujitsu Mobile
- Arrows Tab Wi-Fi FAR70A (Android 3.2)

• iOS
iPhone5 (iOS 6.0.1)
iPhone5s (iOS 7.1)
iPad2 (iOS 7.1.1)
iPad4 (iOS 7.1.1)
iPad mini (iOS 6.0.1, 7.1)

6.

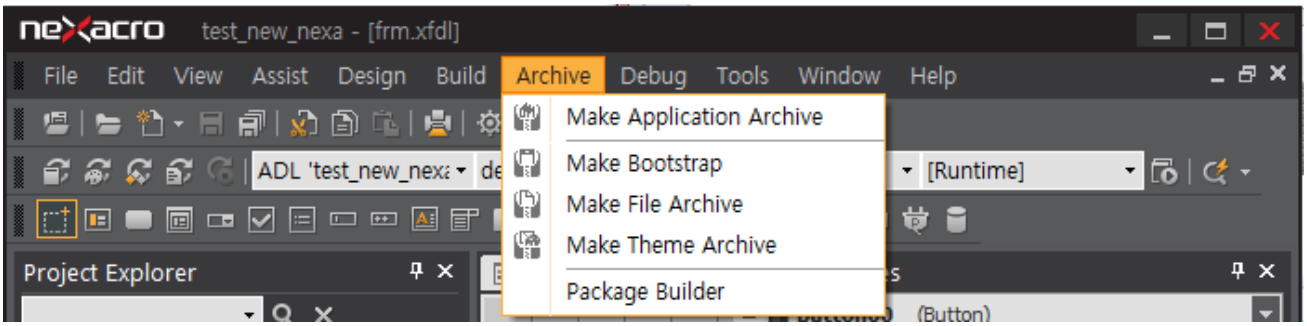
Packaging

Packaging is the process for creating and compressing the information required for execution on a device. It involves creating the *start_android[ios].json* file by using project information, and creating/compressing bootstrap archive files, project archive files and theme archive files.

Packaged files are uploaded to the deploy server and installed by downloading the files from the server.

From Nexacro Studio, access the Archive menu to perform the packaging. Nexacro Studio creates the files to be distributed to the designated paths. Distributing files to the server will be subject to the work environments.

> Use Nexacro Studio to perform the packaging. Using other editors to create or edit packaging might cause an error and is unacceptable.

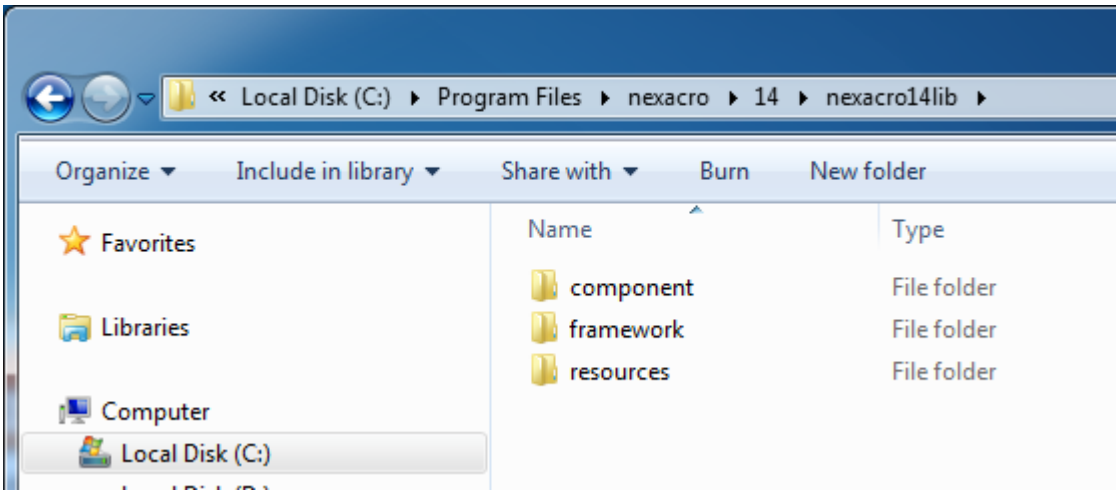


The below table shows the classifications of the files generated through the packaging process. However, you should create a base library by yourself rather than through the packaging process.

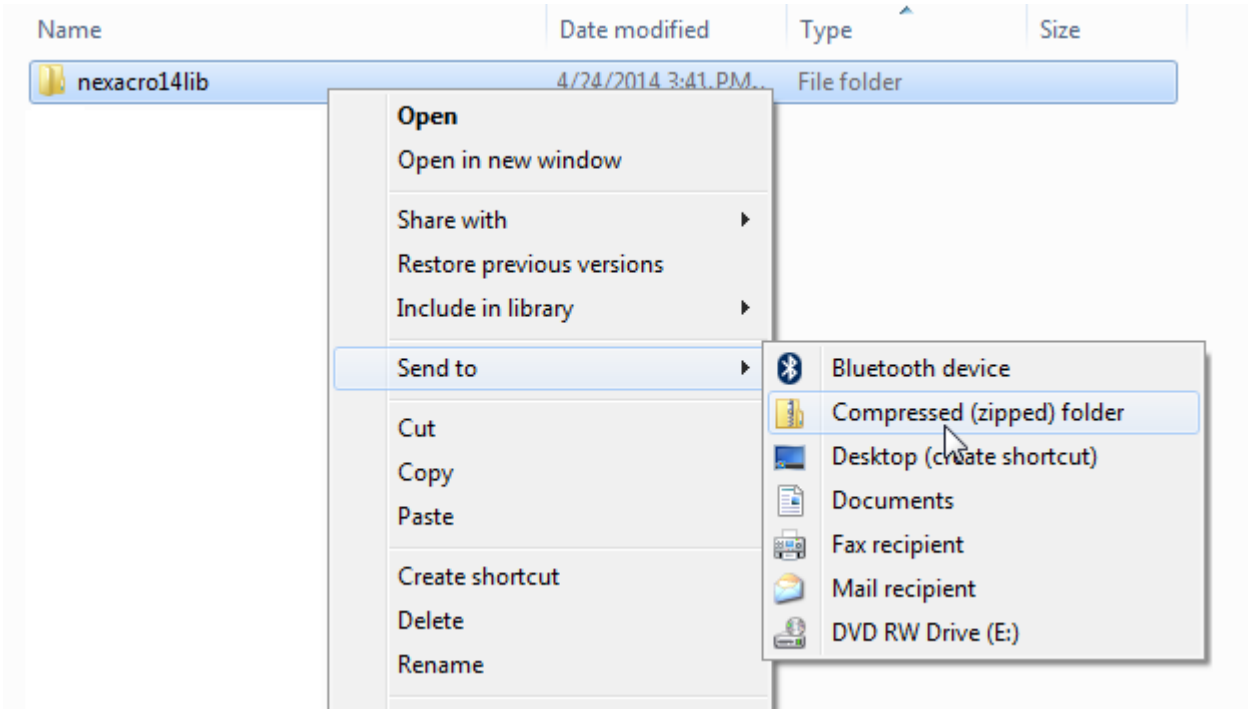
	Application	Bootstrap	File	Theme
Theme file	Y			Y
Executable file	Y	Y		
Form file	Y		Y	
Base library				

6.1 Base Libraries

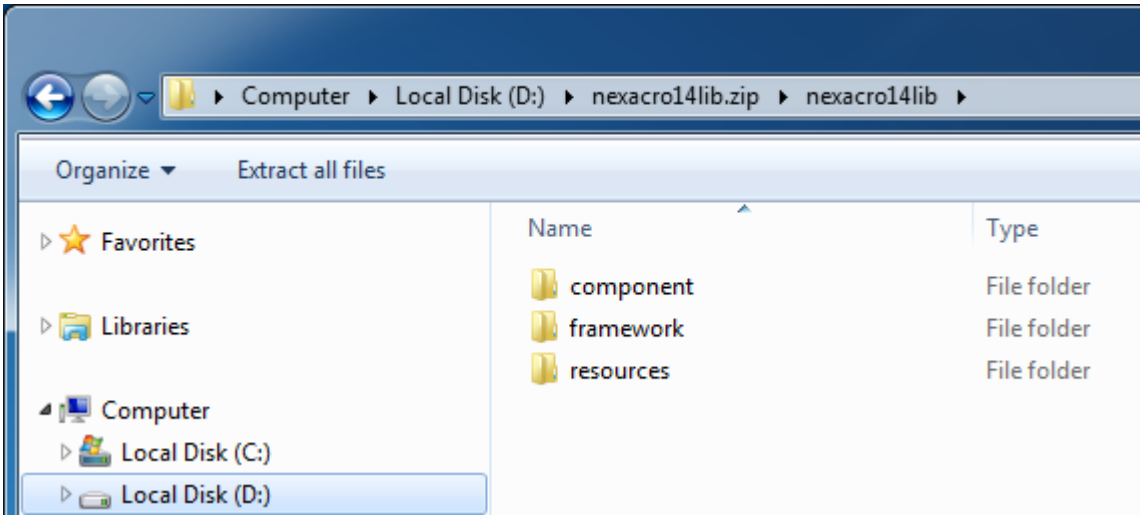
When the Nexacro Platform accesses the installed path, you can see the following folders. When the applications are serviced on the web (when the applications are used on an iOS or Android device), the folders are copied to the corresponding servers as compressed .zip files.



When compressing the folders, copy the nexacro14lib folder to another path and then zip the folder. You can use functions provided with the operating system or separate software, as shown here.



At this moment, the zip file includes the *nexacro14lib* folder.



Depending on software to be used, it is possible that only subfolders - component, framework and resources - are zipped when selecting and zipping the *nexacro14lib* folder. In this case, zip the *nexacro14lib* folder by creating an arbitrary parent folder and creating the zip file.

When creating the Run Template file and selecting the **Merge JSON File** option, select the **Compress JS Files** option and the **Merge JSON File** option using the **[Build > Deploy > Deploy Application]** menu, and use the created *nexacro14lib.zip* file by compressing the folder.

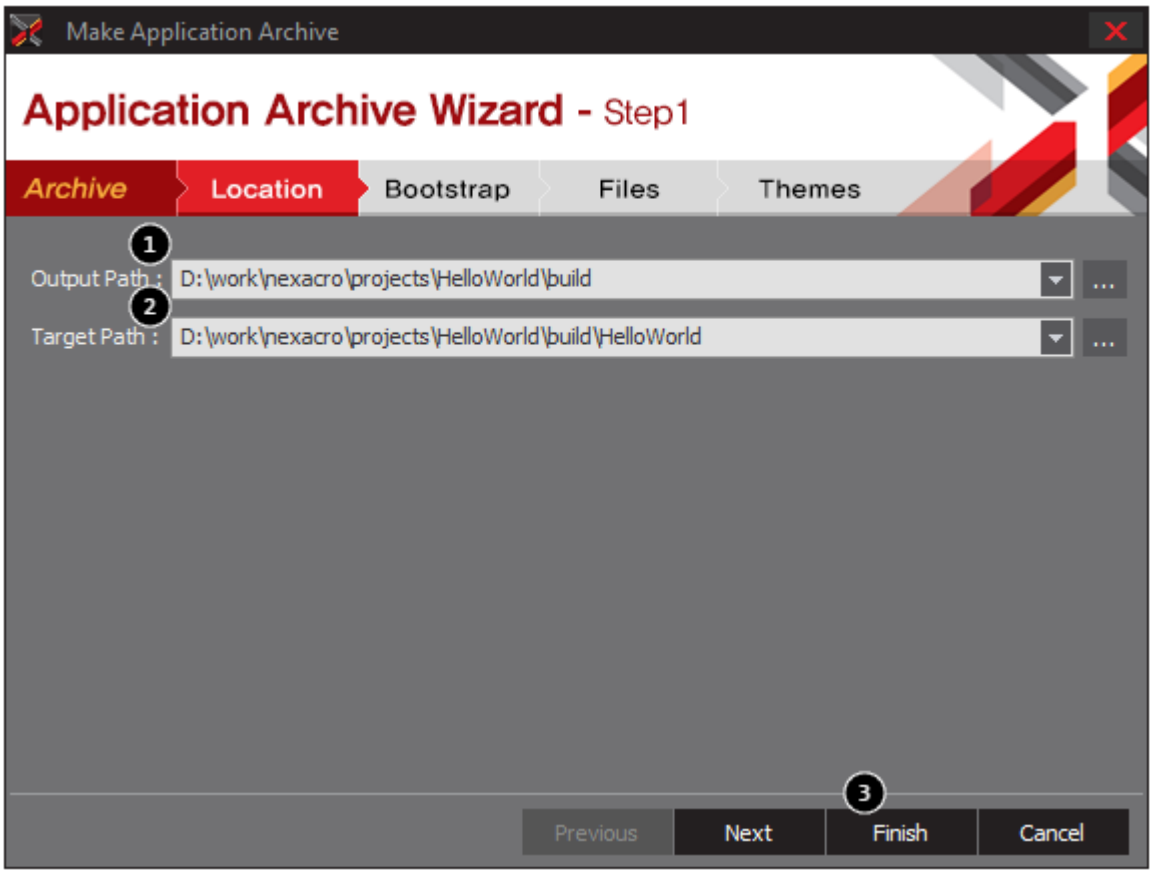
6.2 Archive: Make Application Archive

This is a feature to perform the three aforementioned archiving methods (**Make Bootstrap**, **Make File Archive** and **Make Theme Archive**) at once.

1. Access [**Archive** > **Make Application Archive**] in Nexacro Studio.
2. Proceed with the wizard process by specifying information in the following order.

Step 1

Specify the path for creating archive files as well as the project path.

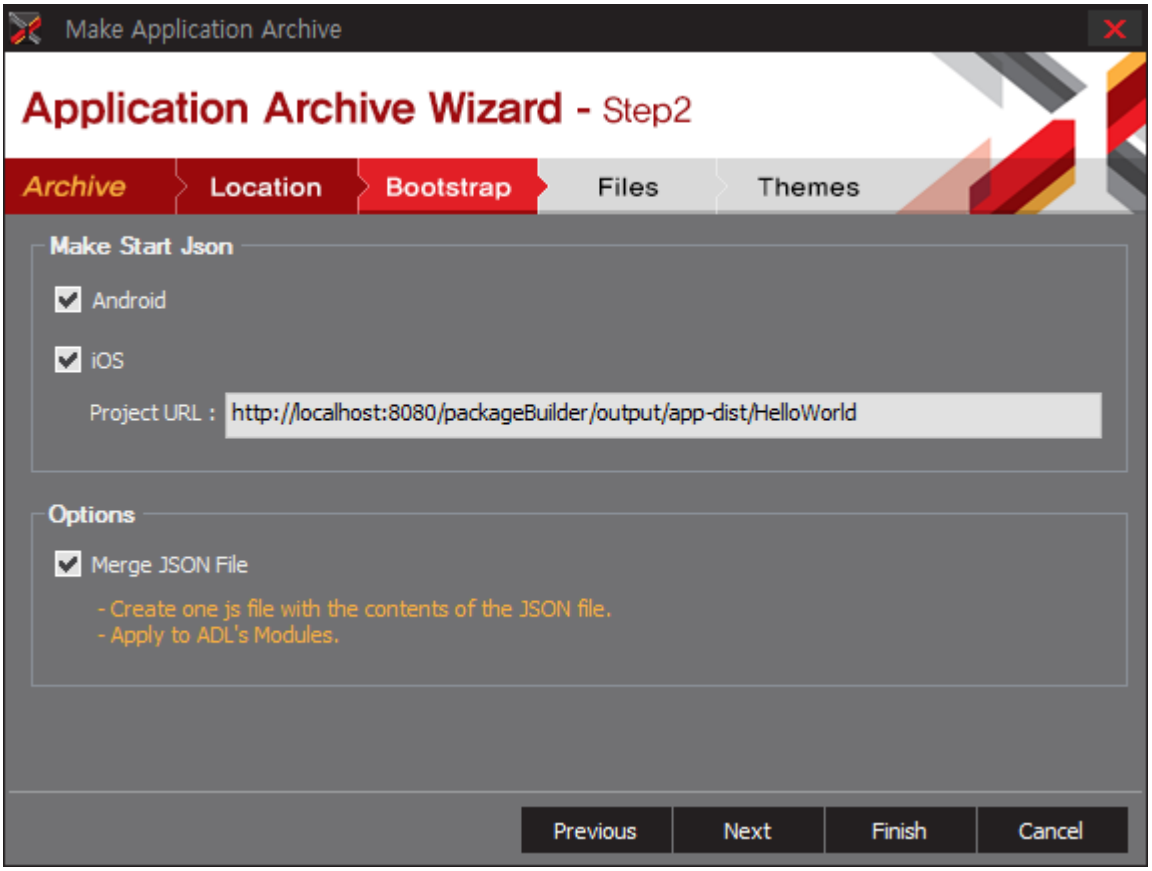


	Item	Description
❶	Output Path	This is a path where archive files are generated.
❷	Target Path	This is a path where a project is compiled. If you have designated a target path for the current project already, the designated path is automatically entered.
❸	Finish	Clicking this button will create archive files with the default values applied. ● *Run.zip* and *start_android[ios].json* files (As for the Merge JSON File option, a previous value will be applied.) ● Project archive file (regenerated after checking the existing project archive file in the output path) ● Theme archive file (Themes that the current ADL file is using are included in the archive file)

Step 2

Specify the options for Bootstrap.

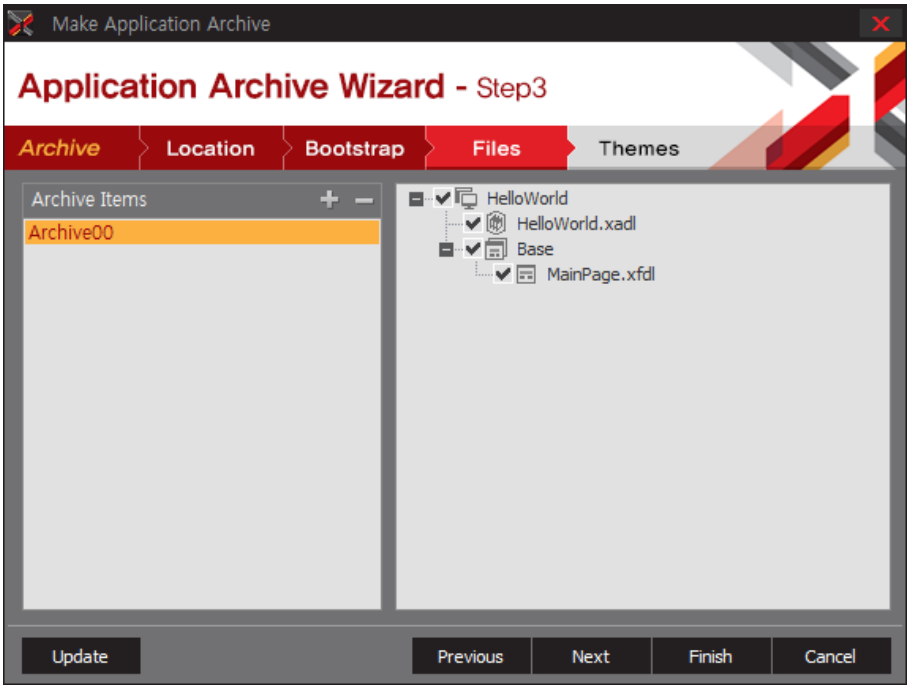
Please refer to the **Make Bootstrap** section for the detailed settings.



Step 3

Specify the options for creating archive files.

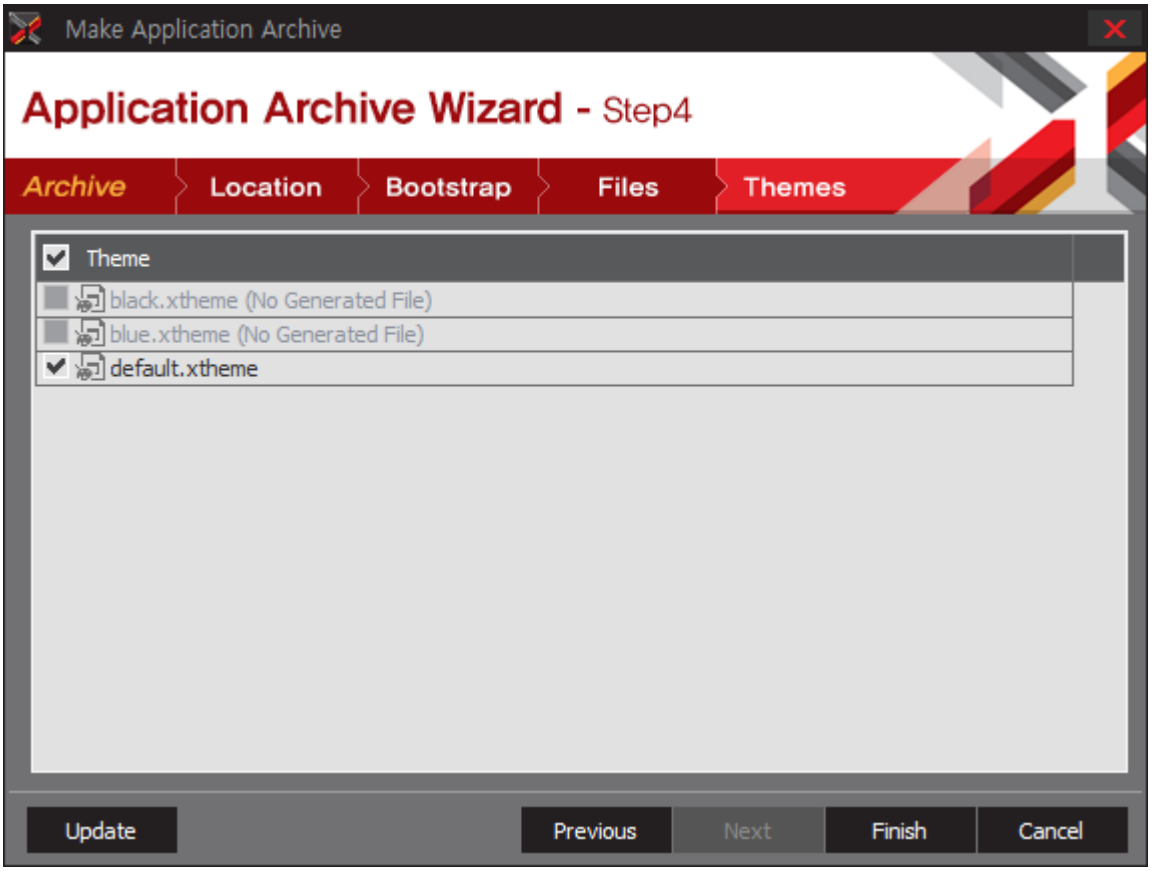
Please refer to the **Make File Archive** section for the detailed settings.



Step 4

Specify the options for incorporating themes into an archive file.

Please refer to the **Make Theme Archive** section for the detailed settings.



3. Apply the generated files to the deployment server.

6.3 Archive: Make Bootstrap

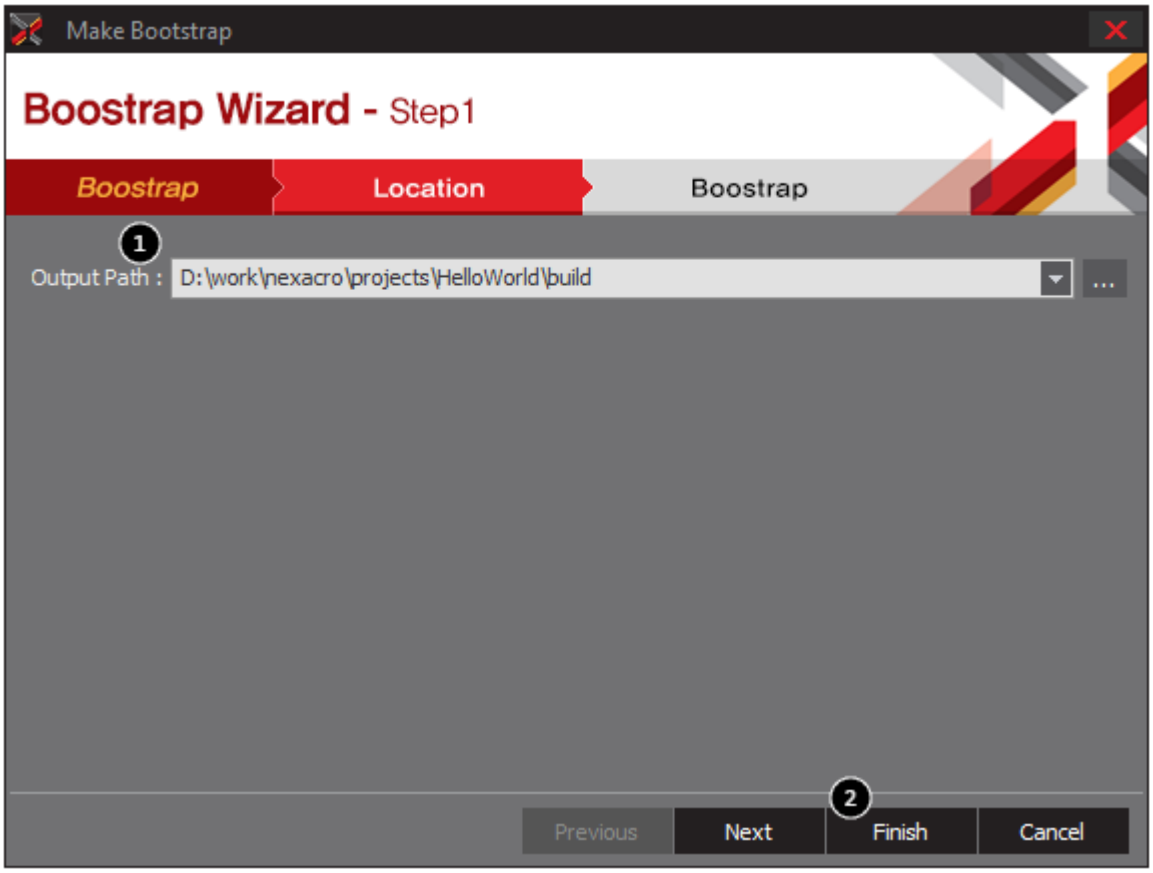
The Make Bootstrap menu option creates files which run apps such as an *start_android[ios].json* and a *Run.zip* including a *Run.html* file.

1. Click **[nexacro studio > Archive > Make Bootstrap]** menu option.

2. Create a *Run.zip* file and an *start_android[ios].json* file as archive files. The *start_android[ios].json* is created based on the information entered in the [Archive > Make File Archive] stage.

Step 1

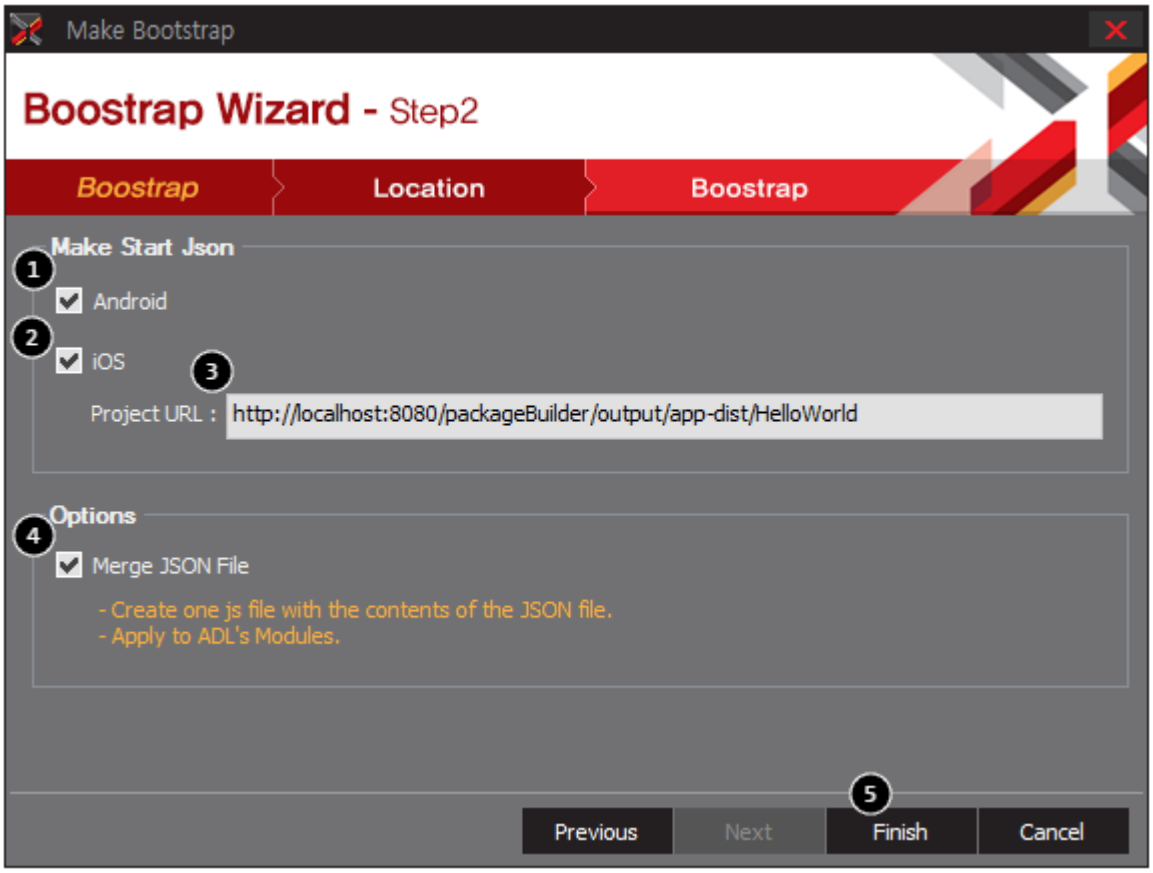
Designate a path where *Run.zip* and *start_android[ios].json* files are generated.



	Item	Description
❶	Output Path	This is a path where *Run.zip* and *start_android[ios].json* files are generated. The information in the **Update** tab in TypeDefinition will be included in *start_android[ios].json*.
❷	Finish	Clicking this button will lead to the generation of *Run.zip* and *start_android[ios].json*. As for the **Merge JSON File** option, a previous value will be applied.

Step 2

Select the mobile platform which your app will be based on, and then specify a project URL. You must select at least one of the two platforms: Android or iOS.



	Item	Description
❶	Android	This option determines whether to create the *start_android.json* file, which is necessary for executing an Android app.
❷	iOS	This option determines whether to create the *start_ios.json* and *Run.zip* files, which are necessary for executing an iOS app.
❸	Project URL	This is a URL where the application is launched. This input box is activated only when ❷ is selected.
❹	Merge JSON File	This option determines the method of calling Nexacro Platform libraries. Before selecting this option, you must select the '**Compress JS file**' and '**Merge JSON file**' options in the [**Build** > **Deploy** > **Deploy Application**] menu when generating a Nexacro application in the form of a JavaScript file.
❺	Finish	Clicking this button will lead to the generation of *Run.zip* and *start_android[ios].json.* *start_android[ios].json* will not be generated if you select neither ❶ nor ❷.

If a local host (127.0.0.1) is set as a URL path in TypeDefinition, the device itself will be acknowledged as the local host instead of the deploy server. For that reason, the URL should be a private IP or URL which the device can access.

3. Create the *start_android[ios].json* file as follows. The contents can be changed depending on the product update.

   When creating the file archive file or the theme archive file, you must create it before creating the *start_android[ios].json* file. This is necessary to be able to use the project/theme archive file offline.

```
{
    "version" : "14.0.0.1",
    "runtime" :
    {
        "target" :
        {
            "os" : "android"
        },
        "framework" :
        [
         "./nexacro14lib/framework/Framework.js",
         "./nexacro14lib/framework/Device.js",
         "./nexacro14lib/framework/Device_Android.js"
        ],
        "component" :
        [
         "./nexacro14lib/component/CompBase.js",
         "./nexacro14lib/component/ComComp.js",
         "./nexacro14lib/component/Grid.js",
         "./nexacro14lib/component/DeviceAPI.js",
         "./archivefiles.js"
        ],
        "form" :
        [

        ],
        "xadl" :
        {
            "launch" : "./HelloWorld.xadl.js",
            "quickview" : "./HelloWorld.xadl.quickview.js"
        },
```

```
        "protocol":
        {
            "adaptors":
            [

            ]
        },
        "resource" :
        {
            "android" : {
                "updateurl" :
"http://172.20.20.20:8080/packageBuilder/output/app-dist/HelloWorld",
                "all" : {
                    "updateurl" : "",
                    "all" : {
                        "updateurl" : "",
                        "resources" :
                        [{"type" : "File", "file" : "Archive00.xzip",
"targetpath" : "", "version" : "1.0", "failpass" : "false"},
                         {"type" : "Engine", "file" : "nexacro14lib.zip",
"targetpath" : "", "version" : "1.0", "failpass" : ""},
                         {"type" : "Theme", "file" : "default.zip",
"targetpath" : "./_theme_", "version" : "1.0", "failpass" : "false"}]
                    }
                }
            },
            "ios" : {
                "updateurl" :
"http://localhost:8080/packageBuilder/output/app-dist/HelloWorld",
                "all" : {
                    "updateurl" : "",
                    "all" : {
                        "updateurl" : "",
                        "resources" :
                        [{"type" : "File", "file" : "Archive00.xzip",
"targetpath" : "", "version" : "1.0", "failpass" : "false"},
                         {"type" : "Engine", "file" : "nexacro14lib.zip",
"targetpath" : "", "version" : "1.0", "failpass" : ""},
                         {"type" : "Engine", "file" : "Run.zip", "targetpath" :
"", "version" : "1.0", "failpass" : ""},
                         {"type" : "Theme", "file" : "default.zip",
```

```
"targetpath" : "./_theme_", "version" : "1.0", "failpass" : "false"}]
                    }
                }
            }
        }
    }
}
```

The *Run.zip* file is not used on the Android OS.
Androids and iOS generate their own necessary files:
- start_android.json
- start_ios.json
When a version value for each file is changed, an app updates the file when running it.

4. Reflect the created file to the deploy server.

6.4 Archive: Make File Archive

Use the **Make File Archive** option from the Archive menu to create a project archive file and register the appropriate information required for distribution in TypeDefinition.

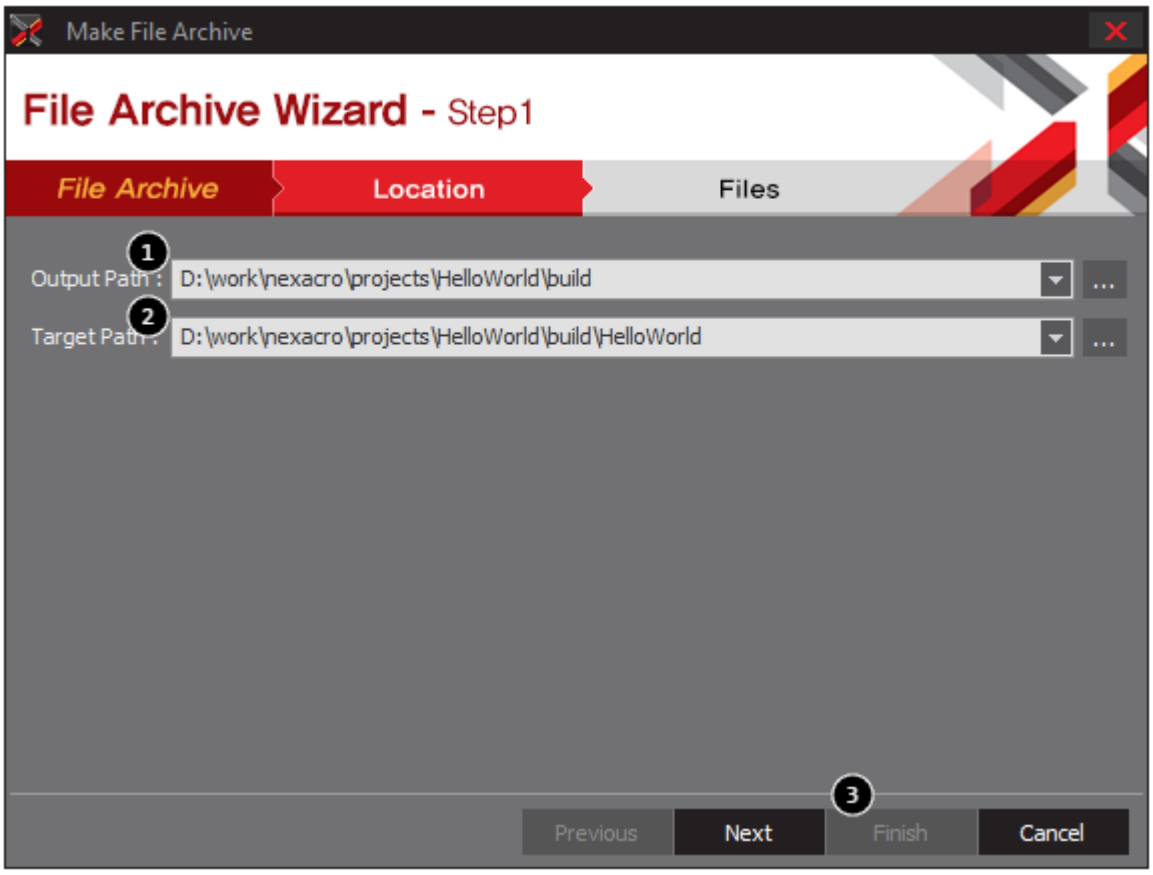
Archive file names are designated by the user and saved with the xzip extension. The project archive file can be downloaded from the deploy server when an app is initially installed. It will be executed by the file saved on the device later on.

1. From the Archive menu, select **[Nexacro Studio > Archive > Make File Archive]**.

2. Organize ADL, forms, style sheets and etc to be included in the archive file, and designate a path where the file will be created.

Step 1

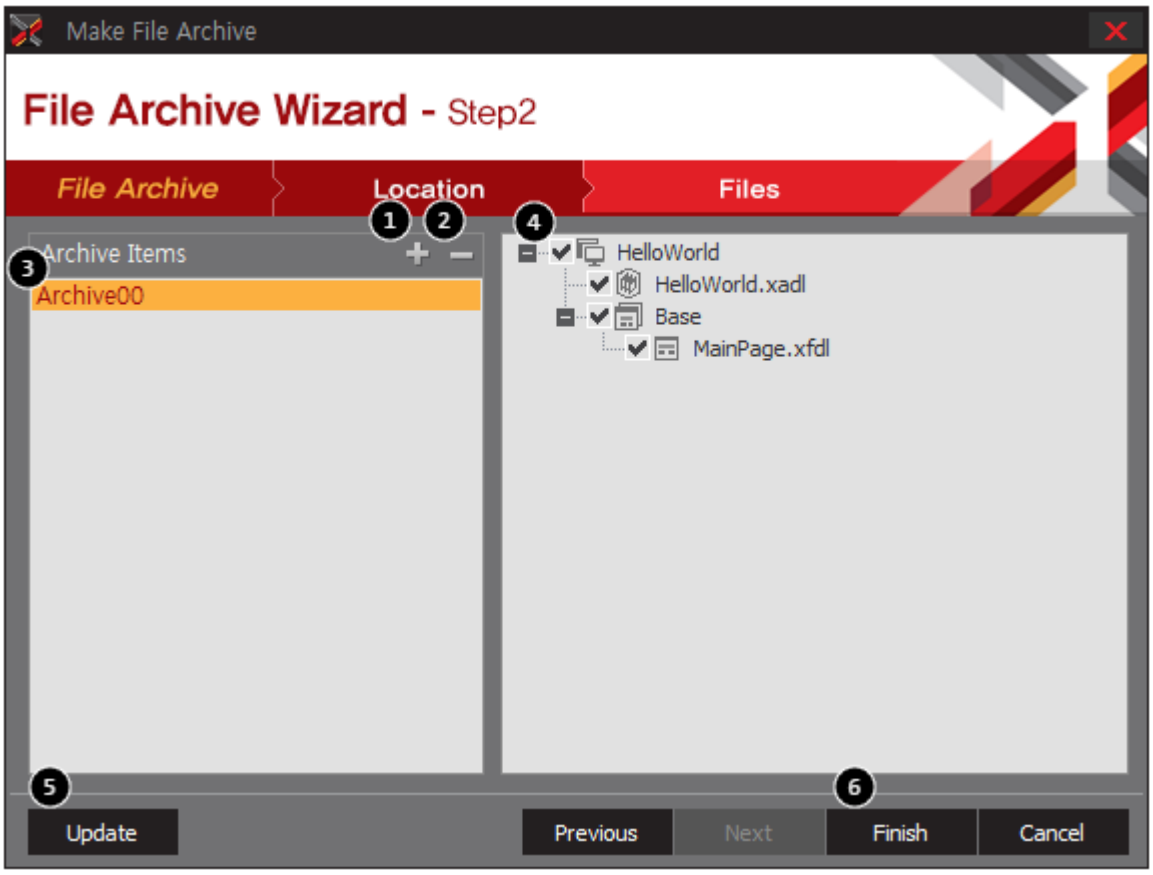
Designate a path where archive files are generated.



	Item	Description
❶	Output Path	This is a path where archive files are generated.
❷	Target Path	This is a path where a project is compiled. If you have designated a target path for the current project already, the designated path is automatically entered.
❸	Finish	Clicking this button will incorporate the project files to an archive file.

Step 2

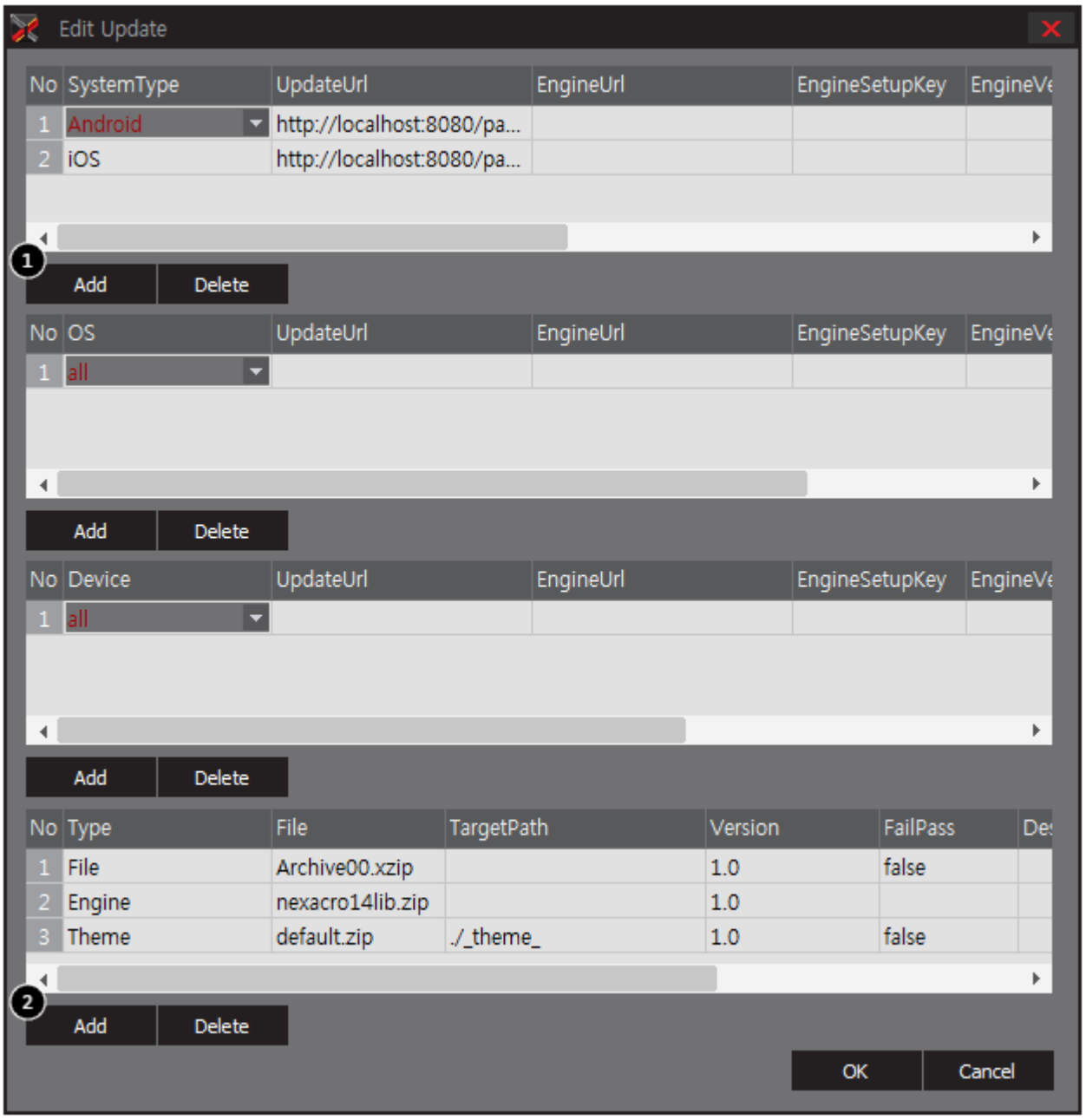
Construct archive files as well as lists of ADL files, forms and style sheets, which are all included in the archive file.



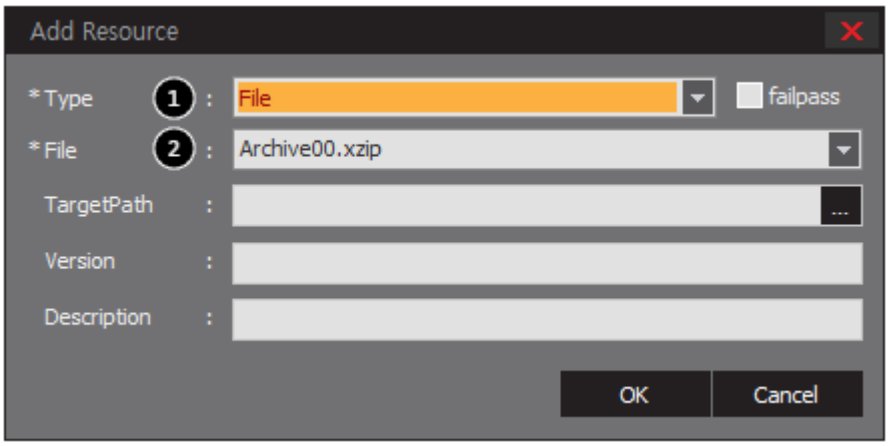
	Item	Description
❶	Add	Adds an archive
❷	Delete	Deletes the selected archive
❸	Archive	This is a list of archive files to create. You must add at least one archive.
❹	Select File	Selects project files to be included in archive files If there are existing archive files, the wizard will check whether there is any overlapping project file. If there are matches, the wizard will show you those projects in the form of a tree. You must select at least one file.
❺	Update	You can add archive files manually to **Update** (Resource) of TypeDefinition.
❻	Finish	If the current archive file is already registered in TypeDefinition, you can skip the Step 3 and finish creating the archive file.

Step 3

You can add archive files to **Update** of TypeDefinition.



	Item	Description
❶	Add	After adding the system type first, you can add information on the OS version, device, and resource. If you need settings of a specific device, you must designate a value corresponding to the value of *system.mobileproducttype* property. ex) SHW-M250S, Nexus One □ You can specify the below property for a detailed setting. - UpdateURL : URL for updating a component library
❷	Add	Register archive files as a Resource. The archive file here refers to the one to be created through the Step 2



	Item	Description
1	Type	You can select the type from **File**, **Engine** and **Theme**. Only the Engine option allows you to download project archives and theme archives from the web server without generating them again.
2	File	You can use the archive file name made in the Step 2 as a file name.

3. Reflect the created archive file in the deploy server manually.

6.5 Archive: Make Theme Archive

Theme archive files are downloaded from the deploy server when apps are initially installed. They will run as data saved on the device later on.

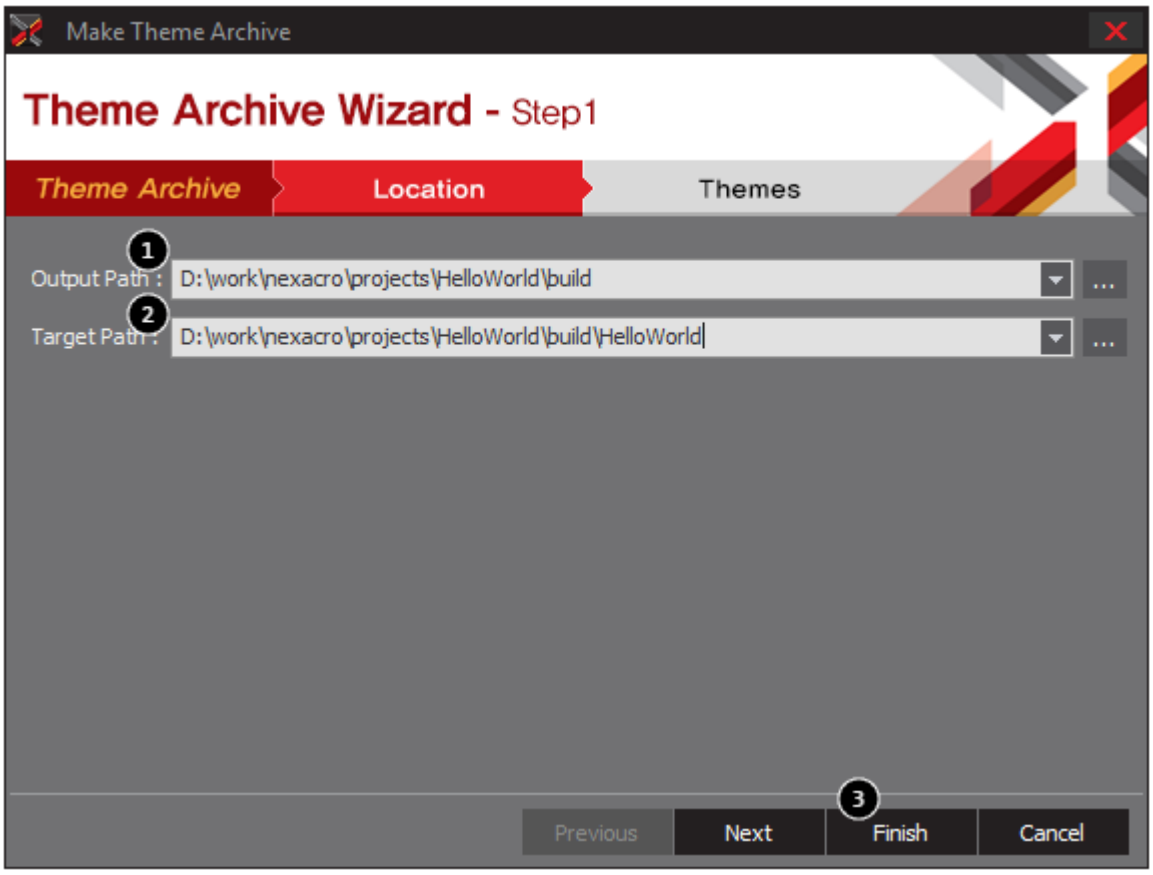
The Make Theme Archive menu option makes archive files with user-designated theme names that will not be changed arbitrarily:

1. Click **[nexacro studio > Archive > Make Theme Archive]** menu.

2. Create a theme archive converted to a JavaScript file.

Step 1

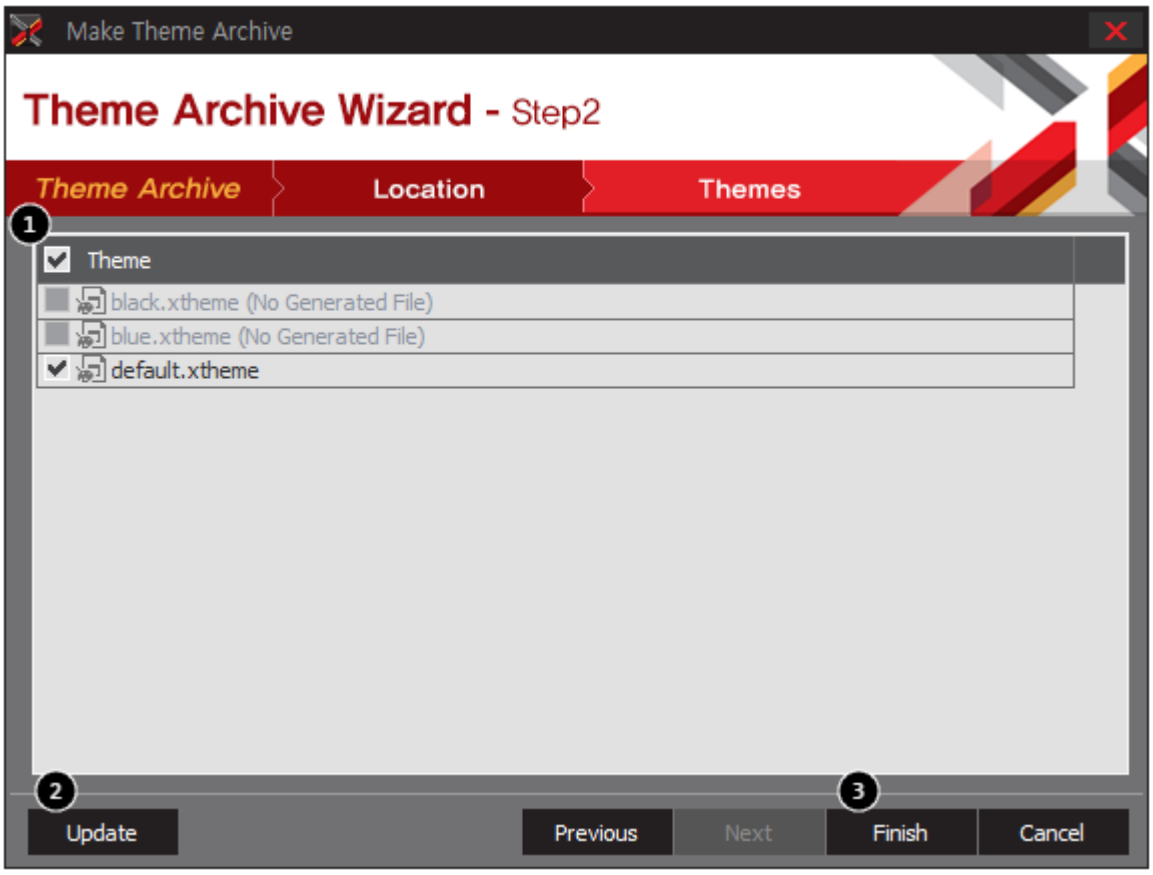
Designate a path where a theme archive file is a generated.



	Item	Description
❶	Output Path	This is a path where an archive file is generated.
❷	Target Path	This is a path where a project is compiled in HTML5. If you designated a path for the current HTML5 project at the time of the compilation, the designated path is automatically entered.
❸	Finish	Clicking this button will generate an archive file that contains the themes applied to the current ADL file.

Step 2

Select themes to be included in an archive file.



	Item	Description
❶	Theme	Select themes to archive. Themes must be complied in HTML5 if they are to be archived. Moreover, those themes must be located in the "**theme**" folder in the path where the theme files are compiled. The wizard shows a list of active themes that are being used in the current project and a list of screen-specific themes. You must select at least one theme.
❷	Update	You can add archive files manually to **Update** (Resource) of TypeDefinition.
❸	Finish	Clicking this button will lead to creating an archive file with selected themes in it.

3. Reflect the created archive file in the deploy server manually.

6.6 Archive: Package Builder

Package Builder is a service that helps a user to develop a mobile hybrid application by building and packaging an application developed through Nexacro Studio.

Nexacro Studio provide the Package Builder wizard ([**Archive > Package Builder**]) to allow users to use Package Builder easily.

Please refer to the Chapter 10. **Building and Packaging App through Package Builder** in this guide book for the details on Package Builder.

7.

App Development and Execution (iOS)

7.1 App Development Environment Configuration

Applications for iOS are developed by using Xcode and iOS SDK on a Mac. Xcode and iOS SDK will not run on a Windows PC. This requires an apple developer account to download Xcode, to create apps and to test them.

This chapter covers the following process. Details can be changed according to updates on the developer's site.

1. Developer Account Registration
2. Xcode and iOS SDK Installation
3. Required Frameworks for Distribution

Developer Account Registration

With an Apple ID, you can download Xcode or use other free services without a developer license.

You can test apps by using a simulator on iOS without going through any authentication process. However, to connect a device you will need to attain the developer license because additional authentication is required.

Apple Account Creation

You need an Apple account first to register a developer account. Register your existing Apple account as a developer account without creating a new developer ID.

You can create a new Apple account using the link below, and you can use the registered email account as the ID. If you have an existing Apple account already, you can skip this step.

https://developer.apple.com/register/createDeveloper.action

Developer License Authorization

If you want to perform simulation testing without connecting to a device, you can register your Apple account as a developer account using the link below. You can download a development Xcode tool or refer to documents including sample codes and videos at iOS Dev Center.

https://developer.apple.com/register
https://developer.apple.com/devcenter/ios

You need to enroll in the iOS Developer Program to download services required for development, test and distribution. It is an annual subscription. Please refer to the link below for the associated services.

https://developer.apple.com/programs/ios/

The Enrollment process will start as soon as you click the **Enroll Now** button at the top of the screen. A developer account can be for either an individual or a company.

Developer
Technologies
Resources
Programs
Support
Member Center
iOS Developer Program
The fastest path from code to customer.
Enroll Now
$99/year
SDK
1. Develop
Develop your application with the iOS SDK and a wealth of technical resources in the iOS Dev Center.
Learn more
2. Test
Test and debug your code on iPad, iPhone and iPod touch to finalize your applications.
Learn more
3. Distribute
Distribute your apps on the App Store and reach millions of iPad, iPhone, and iPod touch users.
Learn more

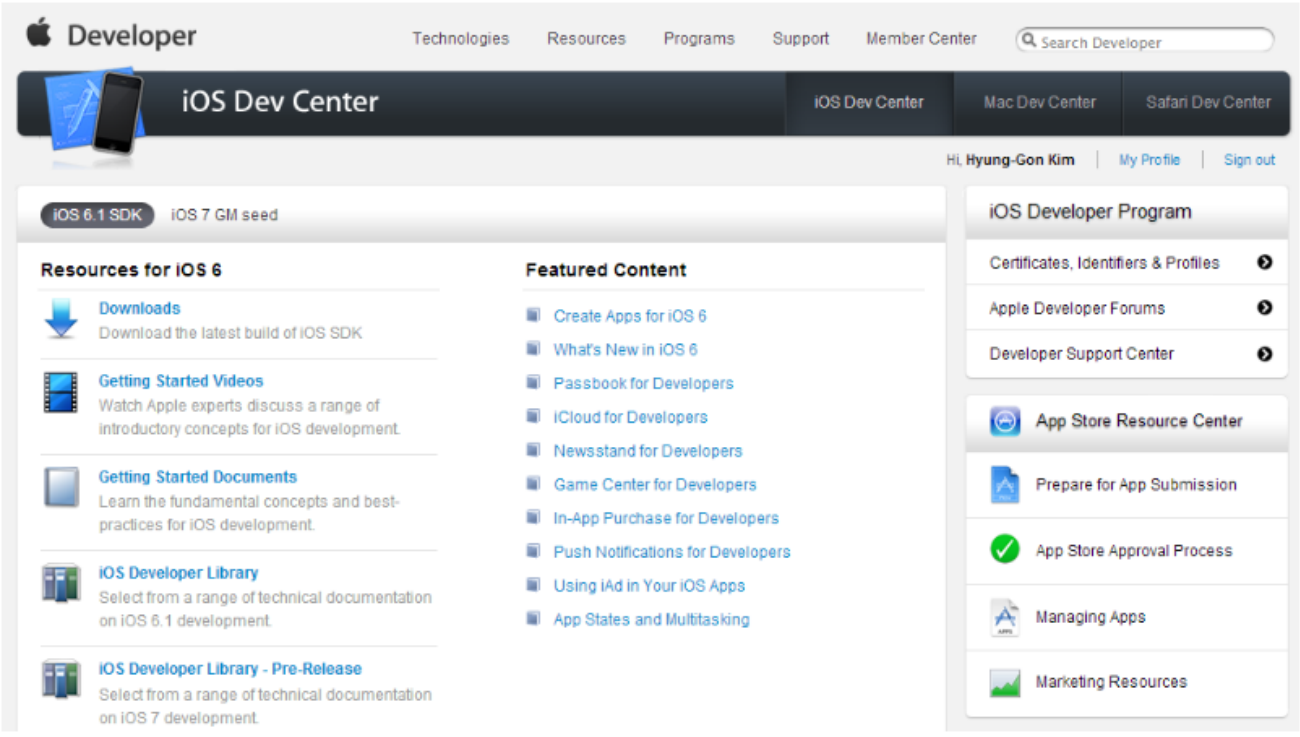
To register the account under the name of a company, you will need a DUNS number, which is used to identify companies all over the world. You can obtain a DUNS number free from the Apple site. After youo apply for the number on the site, you will be requested by a DUNS number-issuing agency to submit certain required documents.

https://developer.apple.com/ios/enroll/dunsLookupForm.action
https://developer.apple.com/support/D-U-N-S/

Once you are enrolled in an account, a person in charge at Apple will examine the enrollment and call you for verification. If there is no problem, you will receive an email with link to a site indicating the enrollment is completed, and you can agree with the license and finally purchase a developer account for a year on the store.

After you complete the purchase, an activation code will be sent to your email. The enrollment process is finally completed when you enter the code on the authentication page.

The iOS Developer Program should be displayed on the right side of the screen of the iOS Dev Center.

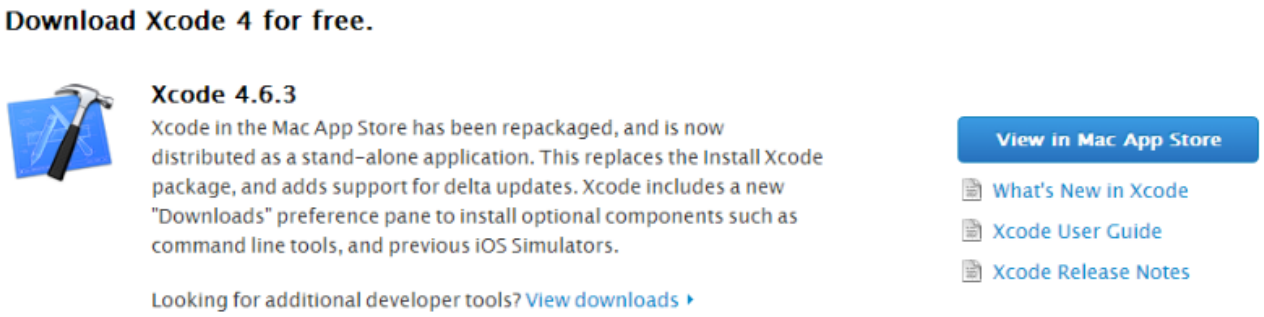


Xcode and iOS SDK Installation

Xcodeis an integrated development environment for the Mac and includes a variety of tools. Even though Xcode is already installed, you might have to update it, depending on iOS SDK versions of connected devices. The versions might support different Xcode versions.

You can check the list of Xcode versions that can be downloaded using the link below. You can also download Xcode directly from the App store. An installation file includes the iOS SDK and Mac SDK.

https://developer.apple.com/xcode/


Download Xcode 4 for free.
Xcode 4.6.3
Xcode in the Mac App Store has been repackaged, and is now distributed as a stand-alone application. This replaces the Install Xcode package, and adds support for delta updates. Xcode includes a new "Downloads" preference pane to install optional components such as command line tools, and previous iOS Simulators.
Looking for additional developer tools? View downloads
View in Mac App Store
What's New in Xcode
Xcode User Guide
Xcode Release Notes

Required Frameworks for Distribution

Several authentication steps are required if you are testing an application while connected to a device. Enrollment in the Developer Program is required for the related frameworks you will need.

Refer to the link below for the required frameworks.

https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html

7.2 App Project Development

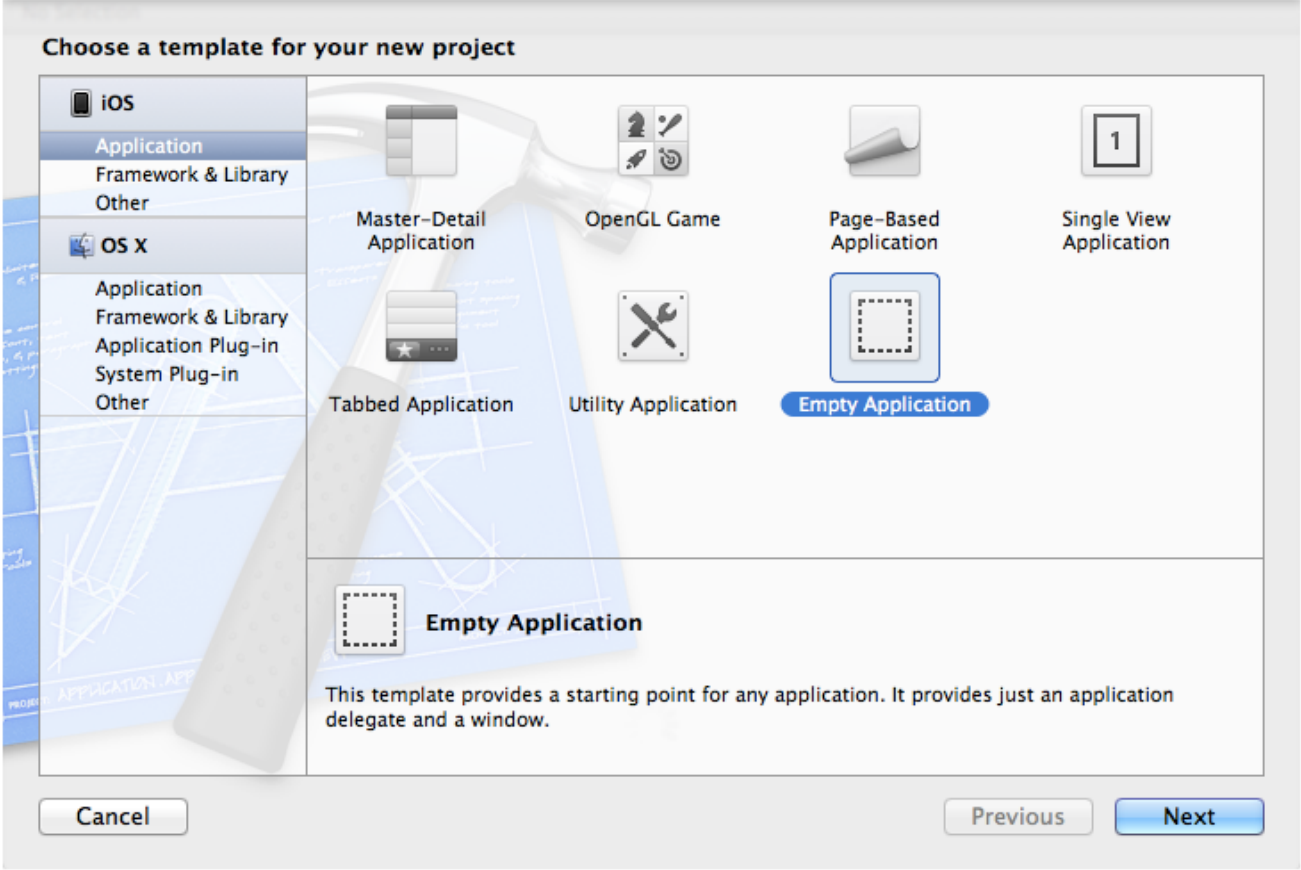
This section goes over the steps for developing application projects for iOS on Xcode. Before starting an app project, ensure that archive files created by an application developed on Nexacro Studio already reside on a designated path.

Creating Projects

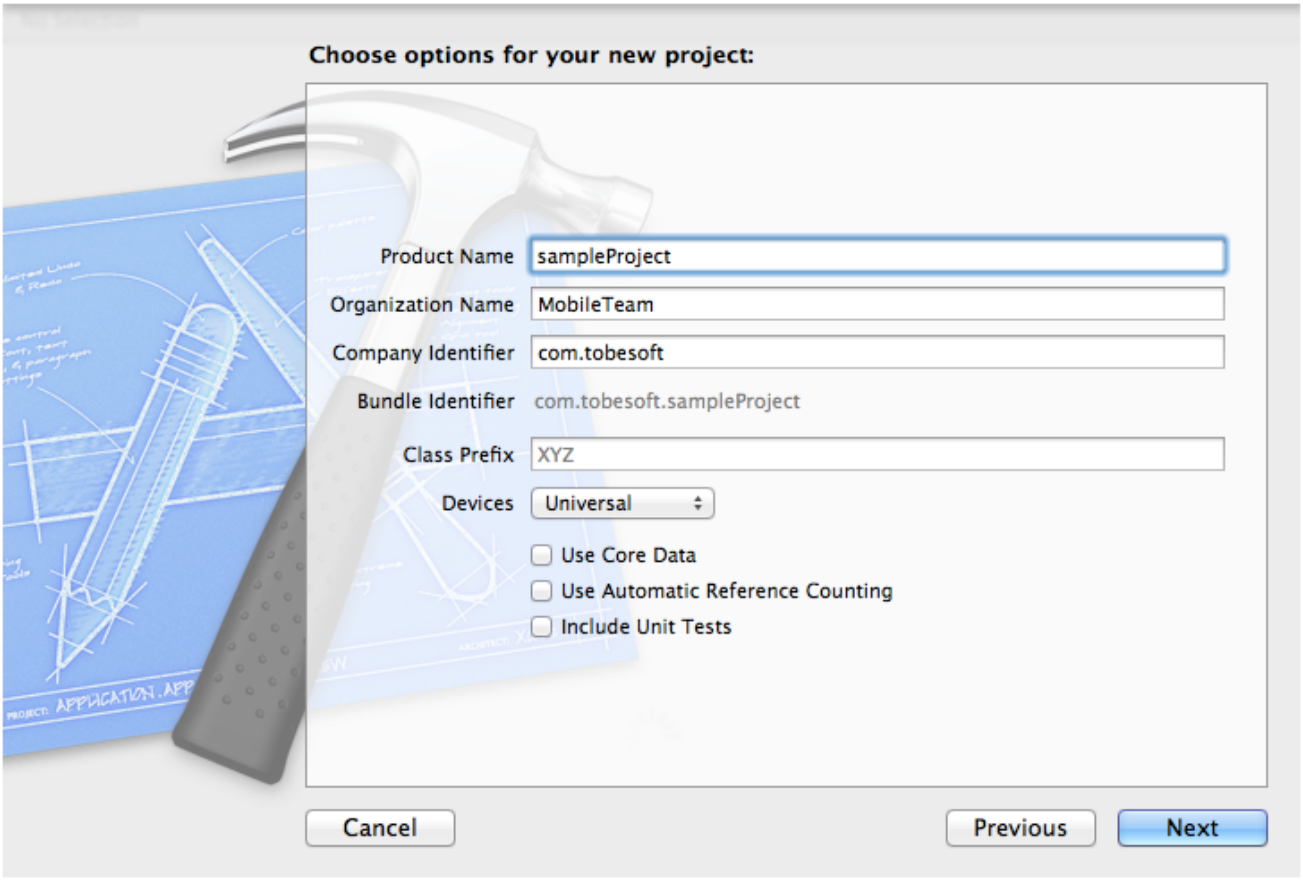
You need to create an iOS project for the applications you develop with Nexacro Platform, and configure the basic environment. Create a new project by selecting:

```
File > New > Project
```

Select the **Empty Application** item when prompted with *Choose a template for your new project*.



Enter the Product Name and other related information. Click the **Create** button after specifying a location where the project will be created.

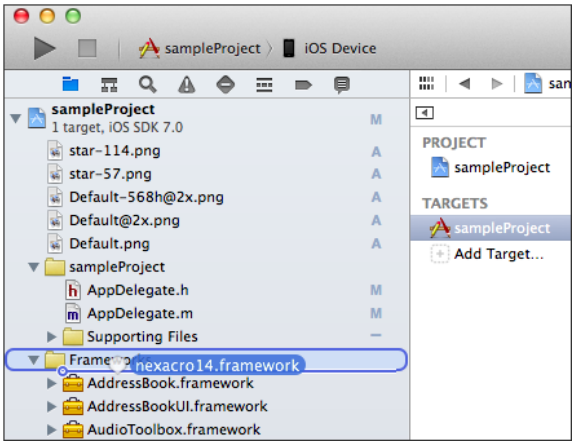


Depending on the version of installed Xcode, creating project methods can differ.

XPLAFORM Library Setting

Nexacro Platform library files are set up to create an optimal environment for your iOS project.

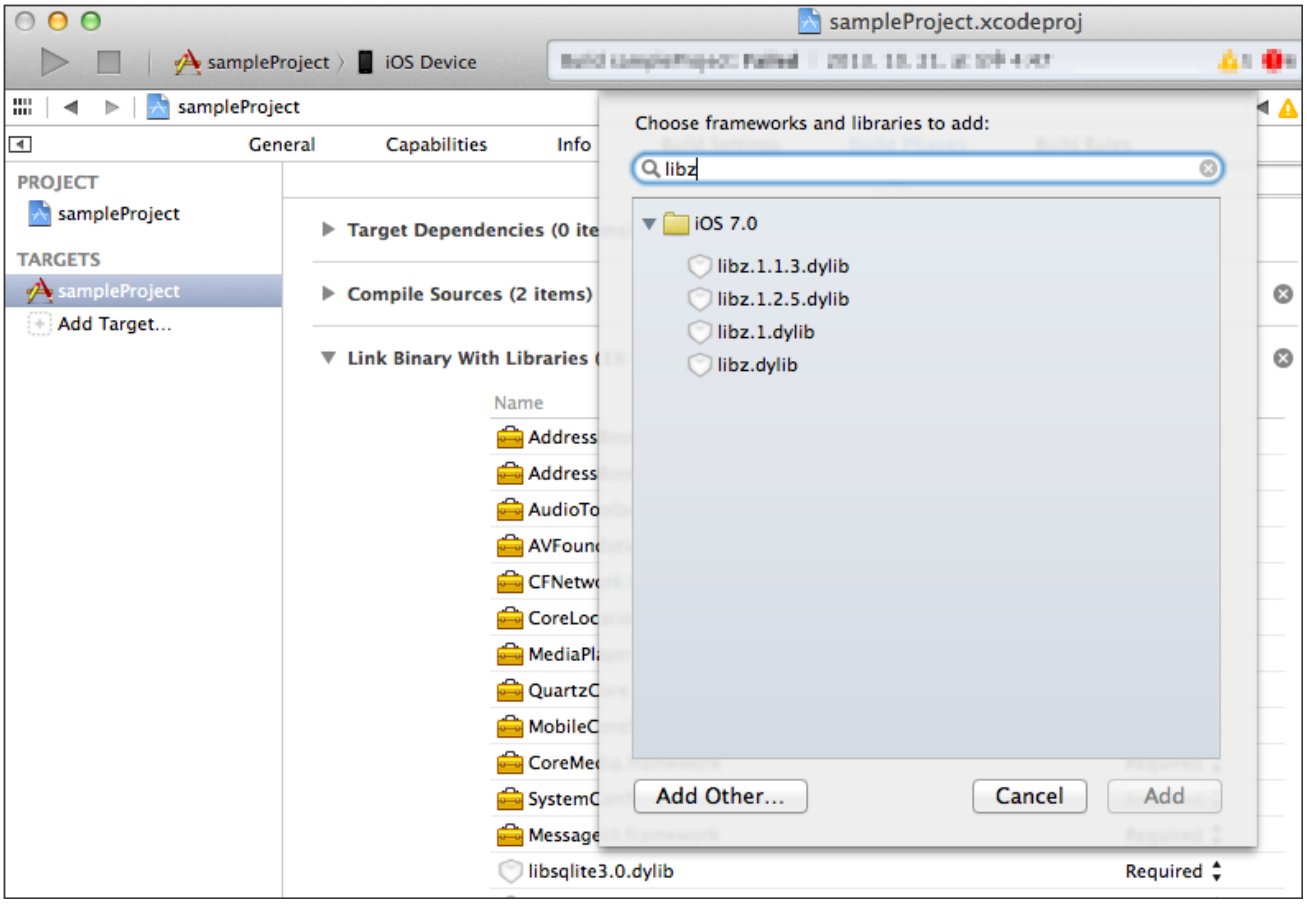
Nexacro Platform libraries are provided in the *nexacro14.framework.zip* file. When unzipped, the libraries are moved to the frameworks of the newly created project.



nexacro14.framework files are provided in two types. When using a simulator, the *i386* file is used. For the iPhone 3GS or later, use the *armv7* file. File settings are different for different product versions.

Additional Library/Framework Settings

You can configure additional libraries and frameworks required for using APIs on iOS. When you select **TARGETS**, you will see *Build Phases* on the top of the screen. Select a relevant tab and then select **Link Binary With Libraries**. You can see three basic libraries as well as added Nexacro Platform libraries.



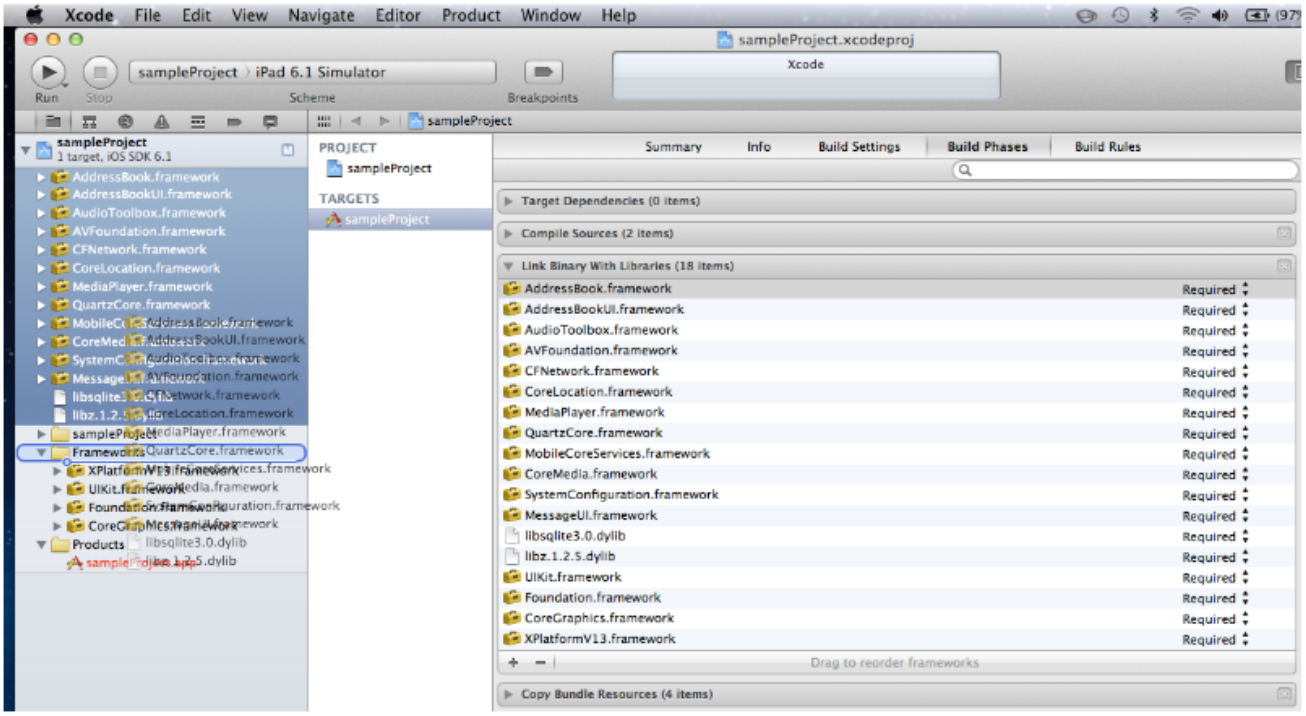
Add 14 items by clicking + button on the button. The items to be added are as follows.

libz.1.2.5.dylib
libsqlite3.0.dylib
MessageUI.fremawork
SystemConfiguration.framework
CoreMedia.framework

MobileCoreServices.framework
QuartzCore.fremawork
MediaPlayer.fremawork
CoreLocation.fremawork
CFNetwork.fremawork
AVFoundation.fremawork
AudioToolbox.framework
AddressBookUI.fremawork
AddressBook.fremawork

The below two items should be added if you use the version released after the begining of June 2015.
Security.framework
libc++.dylib

The libraries/frameworks are added under the project. Select the added items and drag them under the Frameworks folder.



Resource Settings

The various resources to be used with project application must be configured, such as images, icons, messages and layouts.

Resources are organized in a structure similar to the following. However, the image folder can have different structures depending on project settings.

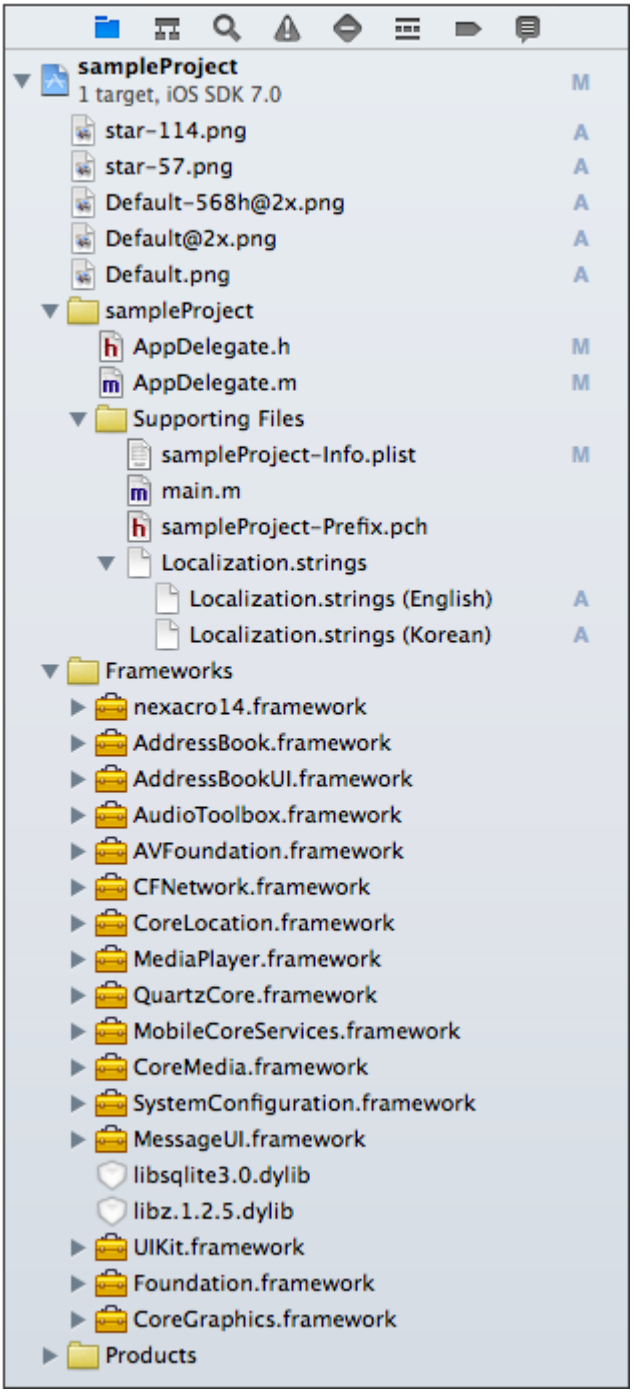
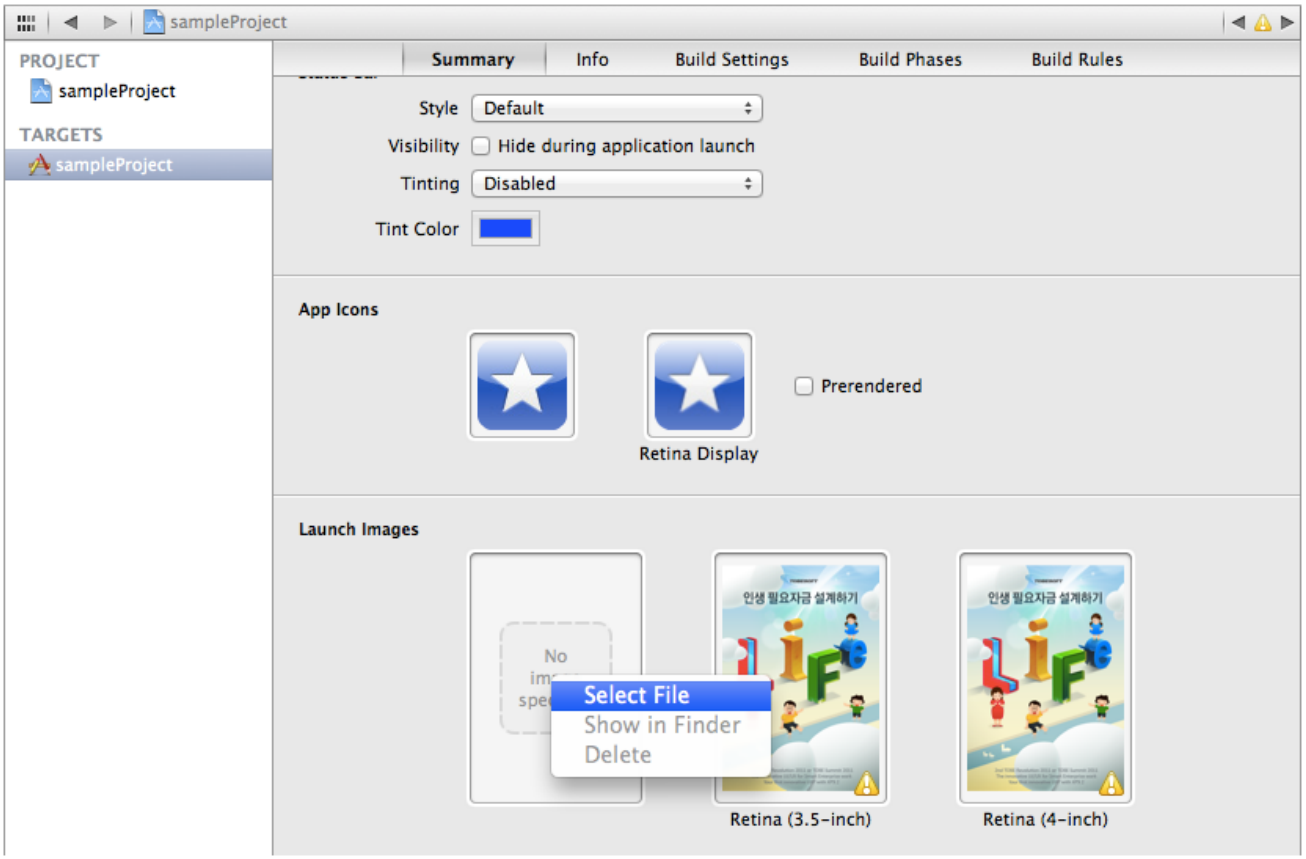


Image Settings

Icons and splash images used on iOS apps are designated respectively. Files which are already made can be imported. When you select the project and then the TARGETS item, you will see the *Summary* tab at the top of the screen showing the App Icons and Launch Images areas.

Initially there are no images displayed. To designate icons and splash images, click **Select File** on the context menu, or drag files to the corresponding screen areas ("App Icons" or "Launch Images").



The files associated with the icons have the existing icon file names. When designating a file, the file cannot be connected to icon if the icon file size is not appropriate. This is because, unlike other custom artwork in your app, the icons and images must meet specific criteria so that iOS can display them properly.

> Please refer to the following Apple Developer Site for details in regard to icon settings.
>
> https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html

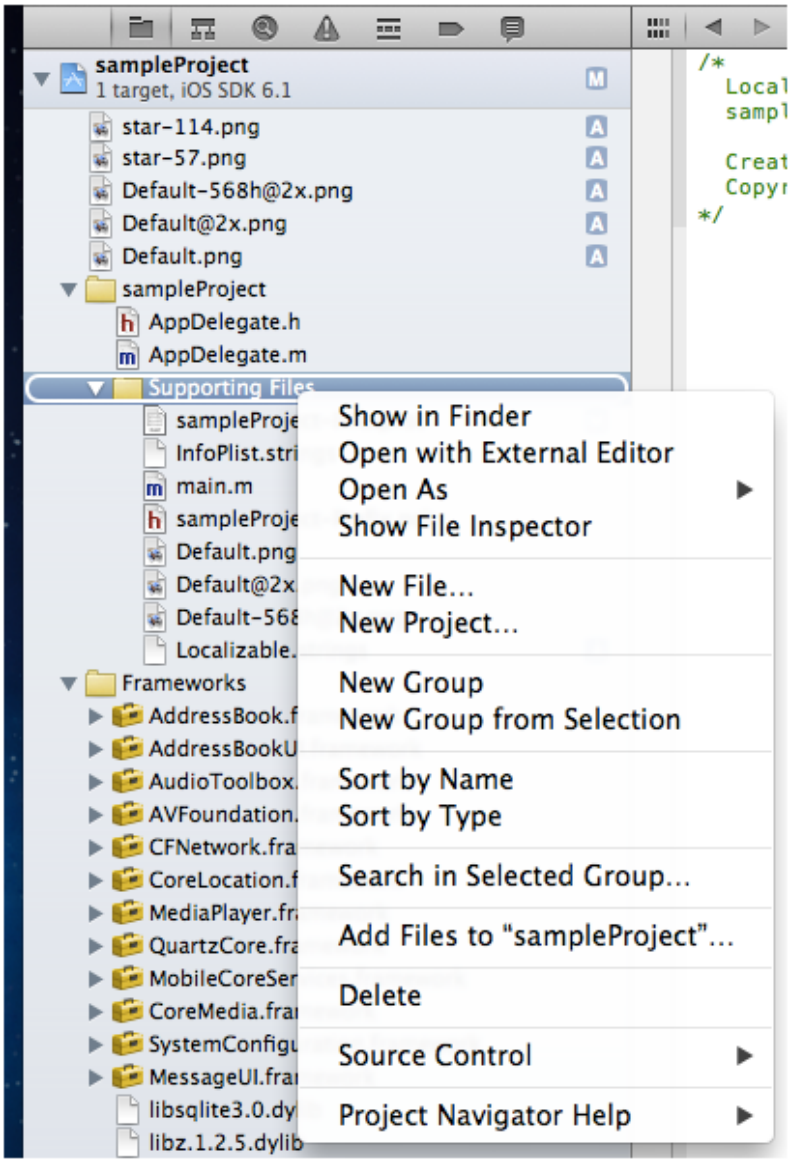
Files connected to splash images are automatically created as *Default.png*, *Default@2x.png* and *Default-586h@2x.png* depending on screen resolution. Do not change the names since they are managed internally.

A file attached with @2x is applied to Retina Display. To support iPhone 5 or later models with optimized images, files attached with -568h@2x are created. Please refer to the Apple Developer site for further details: https://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/App-RelatedResources/App-RelatedResources.html

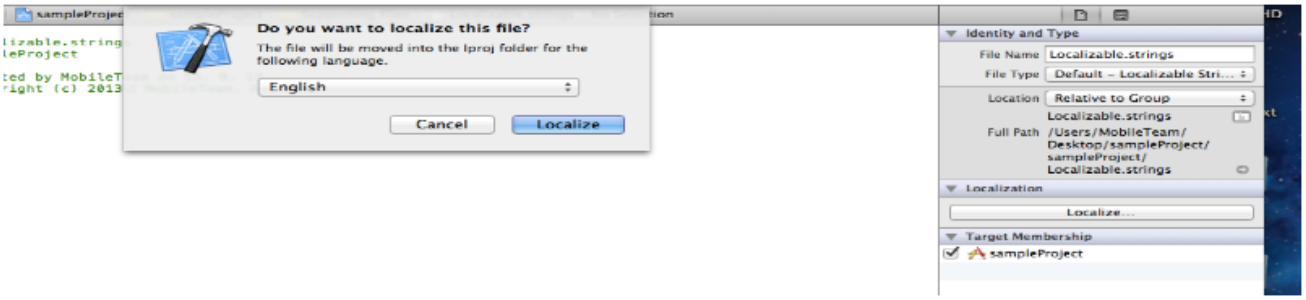
Message Settings

Message setting configures messages used for running apps.

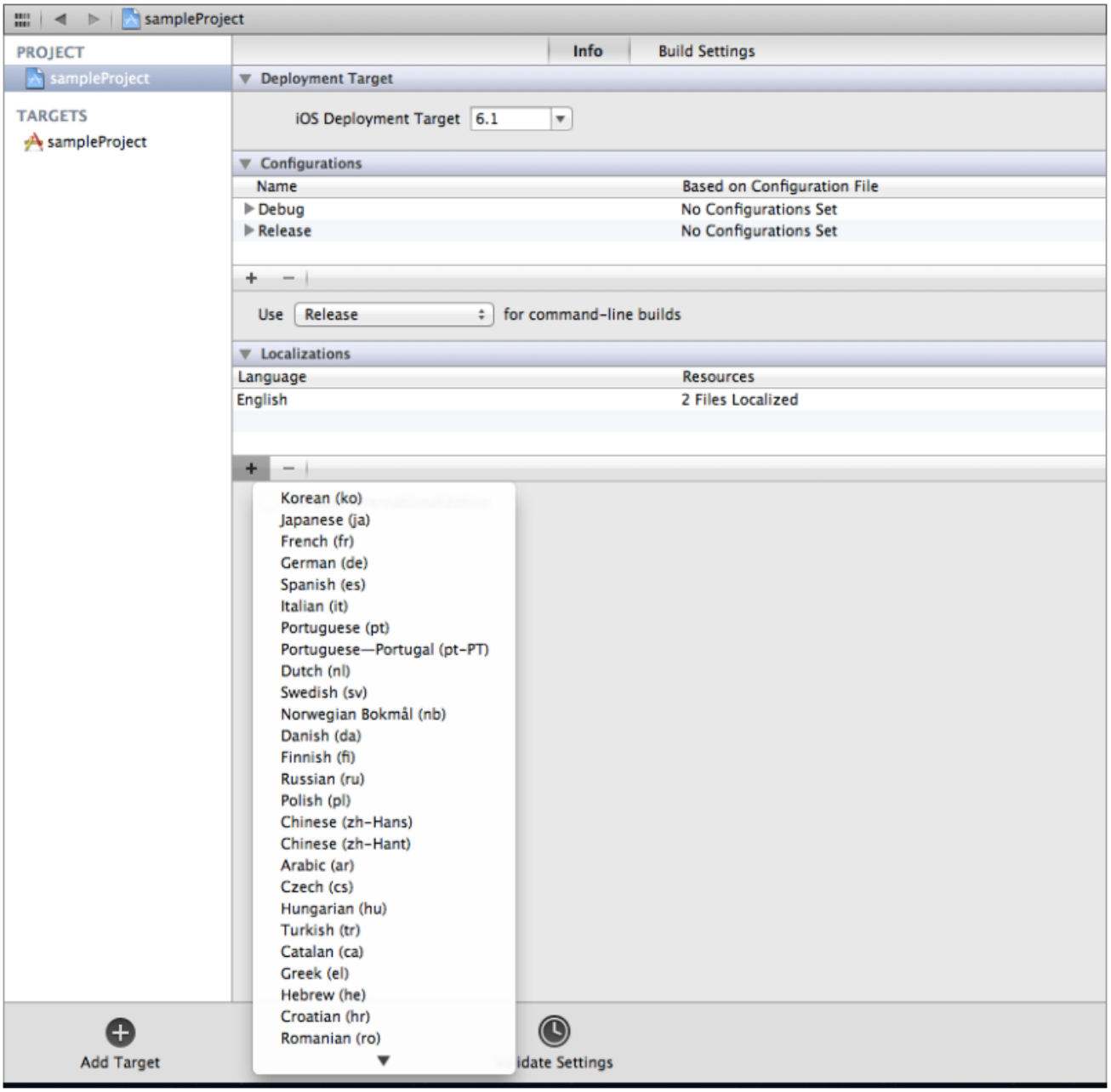
1. Create a new file under the *Supporting Files* folder by selecting the folder and clicking **New File** on the context menu.
2. Select *String File* among the listed templates.
3. Enter the file name as *Localizable*.



4. Select the file and click the **Localize** button in the right window and select your desired language.



Select the project and then the **Info** tab to see the selected language added to the *Localizations* item. You can add languages to the project folder by selecting the **+** button at the bottom.



Revise the created *Localization.strings* file as follows. Revise the added languages appropriately, as well.

```
"needupdate" = "It is need to update.";
"loadingFail" = "First loading is fail.  \r\nPlease restart.";
"updateFail" = "Update is fail.  \r\nPlease restart.";
"notexist" = "Start file is NOT exist.  \r\nPlease restart.";
"BeingUpdated" = "Being updated.";
"wantreplace" = "There is a file of the save name. Do you want to replace? ";
"ok" = "OK";
"cancel" = "Cancel";
"move" = "Move";
"upper" = "Upper";
"filter" = "Filter";
"home" = "Home";
"nofilename" = "No File Name.";
"checkforupdates" = "Check for updates.";
```

Build Environment Setting

Edit the *AppDelegate.h*, *AppDelegate.m* and *main.m* files as suggested here. These were generated when you created the iOS project.

AppDelegate.h

Edit the created application delegate, *AppDelegate.h*, so that it inherits the Nexacro Platform AppDelegate. Revise the related files so that the nexacro14.framework will run.

```
#import <UIKit/UIKit.h>
#import <nexacro14/NexacroAppDelegate.h>
#import <nexacro14/NexacroMainViewController.h>

@interface AppViewController : NexacroMainViewController
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interface
Orientation;
@end

@interface AppDelegate : NexacroAppDelegate <UIApplicationDelegate>
-(NexacroMainViewController*)initMainViewController:(NSInteger)nView;
@end
```

AppDelegate.m

The highlighted part in the *AppDelegate.m* file code below defines a server URL where the *startinfo_ios.json* file created on Nexacro Studio is distributed.

```
#import "AppDelegate.h"

@implementation AppViewController
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interface
Orientation
{
    //return NO;
    return [super shouldAutorotateToInterfaceOrientation:interfaceOrientation];
}
@end

@implementation AppDelegate
-(NexacroMainViewController*)initMainViewController:(NSInteger)nView
{
    [[NexacroResourceManager sharedResourceManager]setBootstrapURL:@"
http://172.10.10.10:8080/nexacro/startinfo_ios.json"];
    return [[AppViewController alloc] initWithLoader:nView];
}
@end
```

main.m

Revise the *main.m* file under the *Supporting Files* folder as follows.

```
#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char *argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

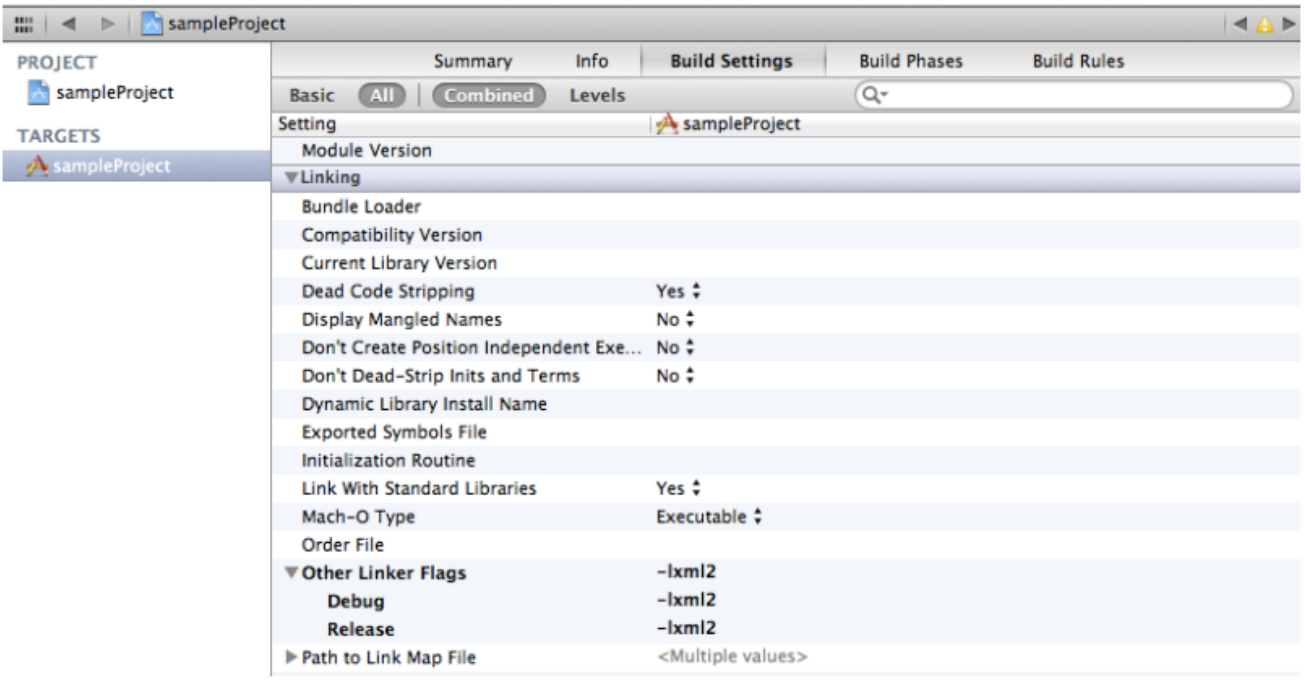
Revise Additional Settings

Settings for some of the items in *Build Settings* should also be revised.

Other Linker Flags

Access **Build Settings > Linking > Other Linker Flags** and set these attribute values:

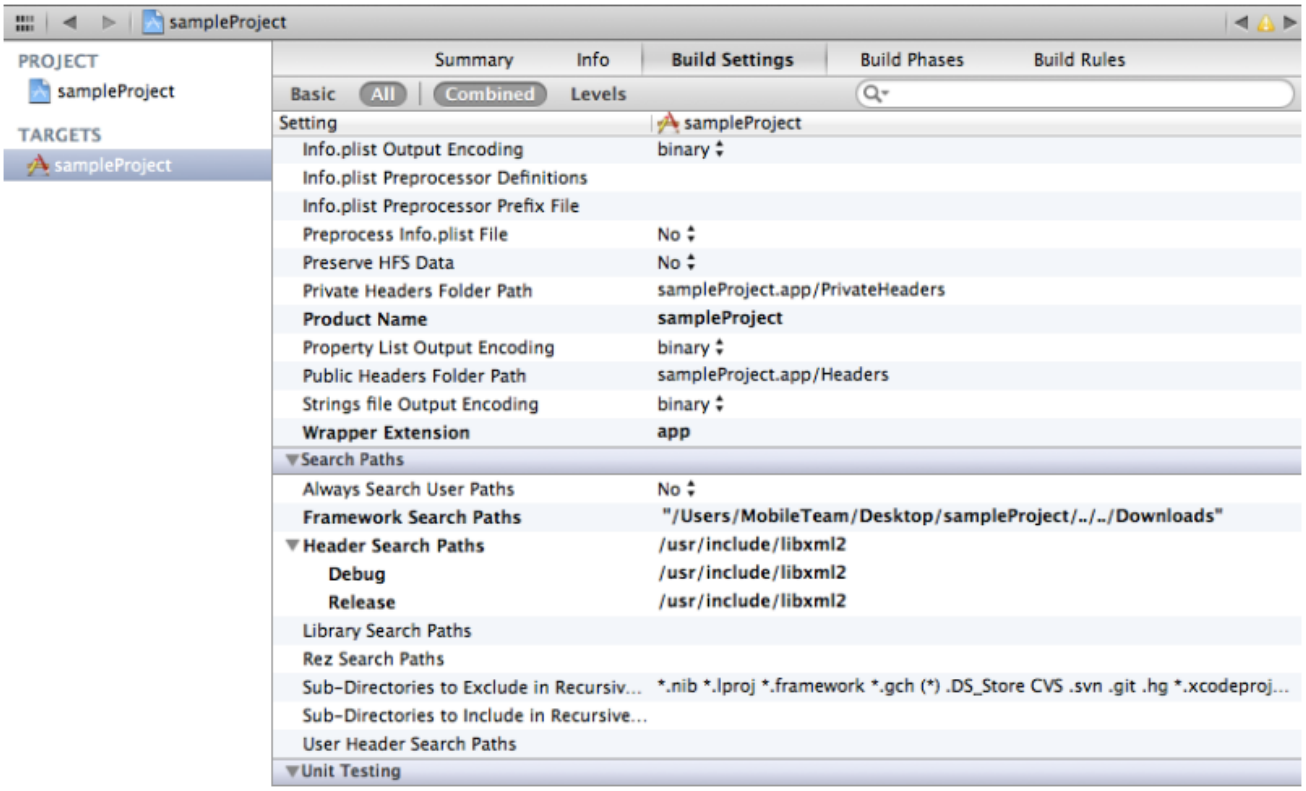
- Debug: -lxml2
- Release: -lxml2



Header Search Paths

Access **Build Settings > Search Paths > Header Search Paths** and set these attributes:

- Debug: /usr/include/libxml2
- Release: /usr/include/libxml2



Build Active Architecture Only

Access **Build Settings > Architectures > Build Active Architecture Only** and modify these attributes:

- Debug: No
- Release: No

Valid Architectures

Access **Build Settings > Architectures > Valid Architectures** and modify these attributes:

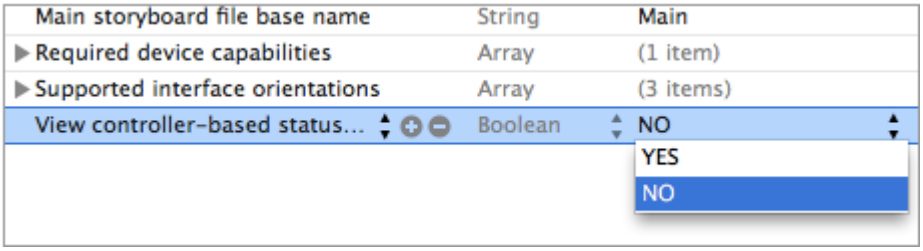
- Debug: armv7 armv7s
- Release: armv7 armv7s

Objective-C Automatic Reference Counting

Access **Build Settings > Apple LLVM5.0-Language > Objective-C Automatic Reference Counting** and set to the value to `No`.

sampleProject-Info.plist File Setting

1. Select the *sampleProject-Info.plist* file under the *Supporting Files* folder.
2. After selecting **View controller-based status bar appearance** instead of *Status bar style* in the property list, , and set the value as **NO**.



iOS8 or later versions are likely to fail to fetch a map if the value of *bUseSendor*, a parameter of the Map compoment's *load* method, is set to "true." In such a case, select the *Info.plist* file and add the following items.

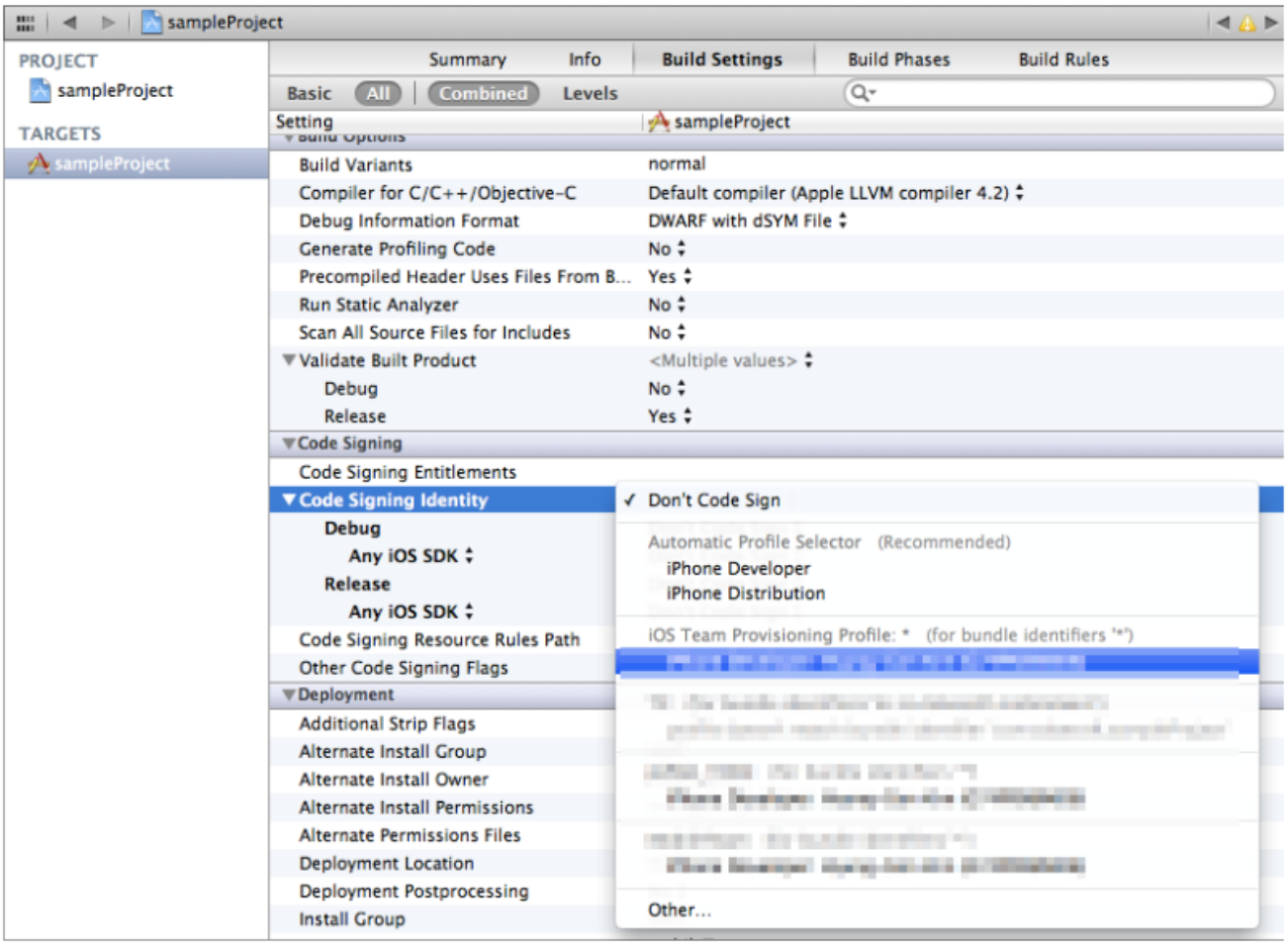
- `Privacy - Location Usage Description` (You need to select this item from the existing property list in the file. This item is used to specify a message displayed when an app first executes GPS.)
- `NSLocationAlwaysUsageDescription` (You need to add this item to the plist file by specifying the Type as "String")
- `NSLocationWhenInUseUsageDescription` (You need to add this item to the plist file by specifying the Type as "String")

The aforementioned settings are intended to meet the strengthened security demands in iOS8 or later for using GPS. Those settings are also required for other GPS-linked components (e.g. Geolocation).

Code Signing

When you use a simulator for testing, there is no need for authentication. However, an additional setting is required to connect to an actual device.

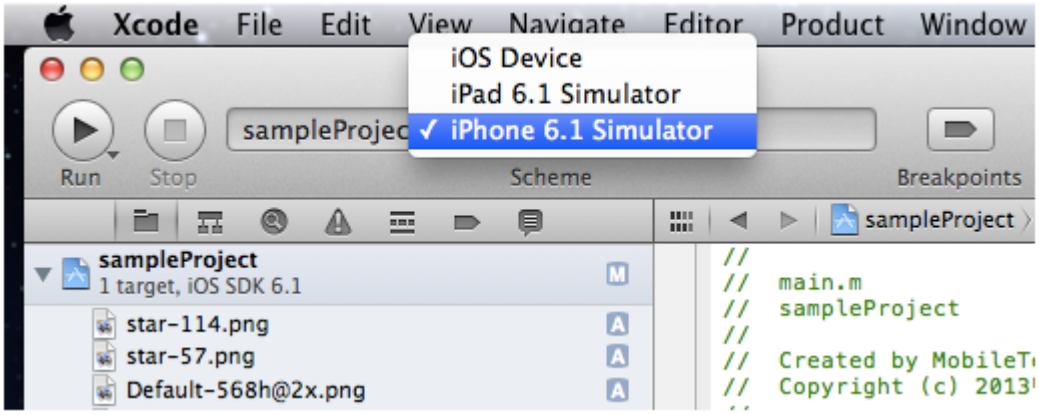
Set the **Code Signing Identity** property in *Code Signing* with an appropriate profile.


sampleProject
PROJECT
sampleProject
TARGETS
sampleProject
Summary
Info
Build Settings
Build Phases
Build Rules
Basic
All
Combined
Levels
Setting
sampleProject
Build Variants
normal
Compiler for C/C++/Objective-C
Default compiler (Apple LLVM compiler 4.2)
Debug Information Format
DWARF with dSYM File
Generate Profiling Code
No
Precompiled Header Uses Files From B...
Yes
Run Static Analyzer
No
Scan All Source Files for Includes
No
Validate Built Product
<Multiple values>
Debug
No
Release
Yes
Code Signing
Code Signing Entitlements
Code Signing Identity
Debug
Any iOS SDK
Release
Any iOS SDK
Code Signing Resource Rules Path
Other Code Signing Flags
Deployment
Additional Strip Flags
Alternate Install Group
Alternate Install Owner
Alternate Install Permissions
Alternate Permissions Files
Deployment Location
Deployment Postprocessing
Install Group
Don't Code Sign
Automatic Profile Selector (Recommended)
iPhone Developer
iPhone Distribution
iOS Team Provisioning Profile: * (for bundle identifiers '*')
Other...

7.3 App Test

You can test an application directly by connecting the iOS device to a PC using a USB cable, or by connecting the device to a simulator.

1. Install an appropriate profile on the device by selecting the target.



Execute the test Xcode by clicking the run icon on the top of the screen or accessing **Product > Run**.

8.
App Development and Execution (Android)

8.1 App Development Environment Configuration

Nexacro Platform Android apps are developed using Eclipse and the Android SDK. Basic processes are the same as developing Android apps.

Refer to the Google Android webpage for Eclipse and Android SDK installation files and related data.

http://developer.android.com/sdk/installing.html

Since updated versions for Eclipse and Android SDK are released often, you need to install the latest development environment using the above link. Details regarding the installation can differ depending on the updates to Eclipse, ADT Plug-in, and Android SDK.

The Android development environment is configured using the following process:

1. JDK(Java SE Development Kit) installation
2. Eclipse installation
3. Android SDK installation
4. ADT plug-in installation

Because the app development environment is already configured, there is no need to go over it here.

JDK(Java SE Development Kit) Installation

Because Eclipse is based on Java, you need to configure the JDK (Java SE Development Kit) environment. You can download and install the JDK for free.

Download

Download the JDK for your system environment using the site below.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Java SE Development Kit 7u17

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

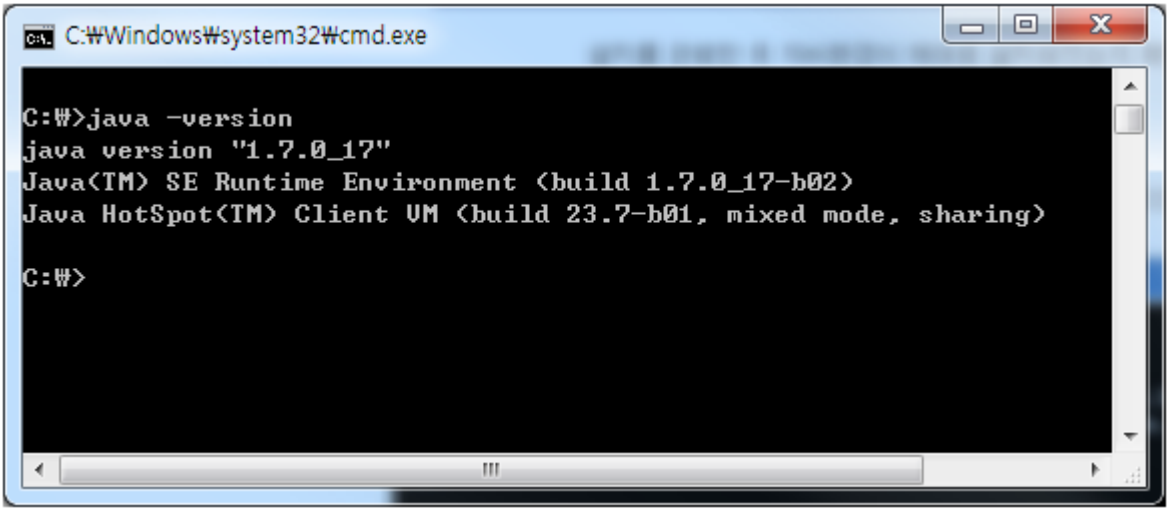
Product / File Description	File Size	Download
Linux x86	106.65 MB	jdk-7u17-linux-i586.rpm
Linux x86	92.97 MB	jdk-7u17-linux-i586.tar.gz
Linux x64	104.78 MB	jdk-7u17-linux-x64.rpm
Linux x64	91.71 MB	jdk-7u17-linux-x64.tar.gz
Mac OS X x64	143.78 MB	jdk-7u17-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.39 MB	jdk-7u17-solaris-i586.tar.Z
Solaris x86	91.67 MB	jdk-7u17-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.92 MB	jdk-7u17-solaris-sparc.tar.Z
Solaris SPARC	95.32 MB	jdk-7u17-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.97 MB	jdk-7u17-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.59 MB	jdk-7u17-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.61 MB	jdk-7u17-solaris-x64.tar.Z
Solaris x64	15.02 MB	jdk-7u17-solaris-x64.tar.gz
Windows x86	88.75 MB	jdk-7u17-windows-i586.exe
Windows x64	90.42 MB	jdk-7u17-windows-x64.exe

In this chapter, the *jdk-7u17-windows-i586.exe* (version 1.7.0_17) file is used. Some of this information may differ based on the version you are installing.

Installation Check

When you run the installation file, it automatically executes the necessary environment settings and installation process. Check that it is successfully installed in the command window after the installation is completed by entering this command:

```
java -version
```



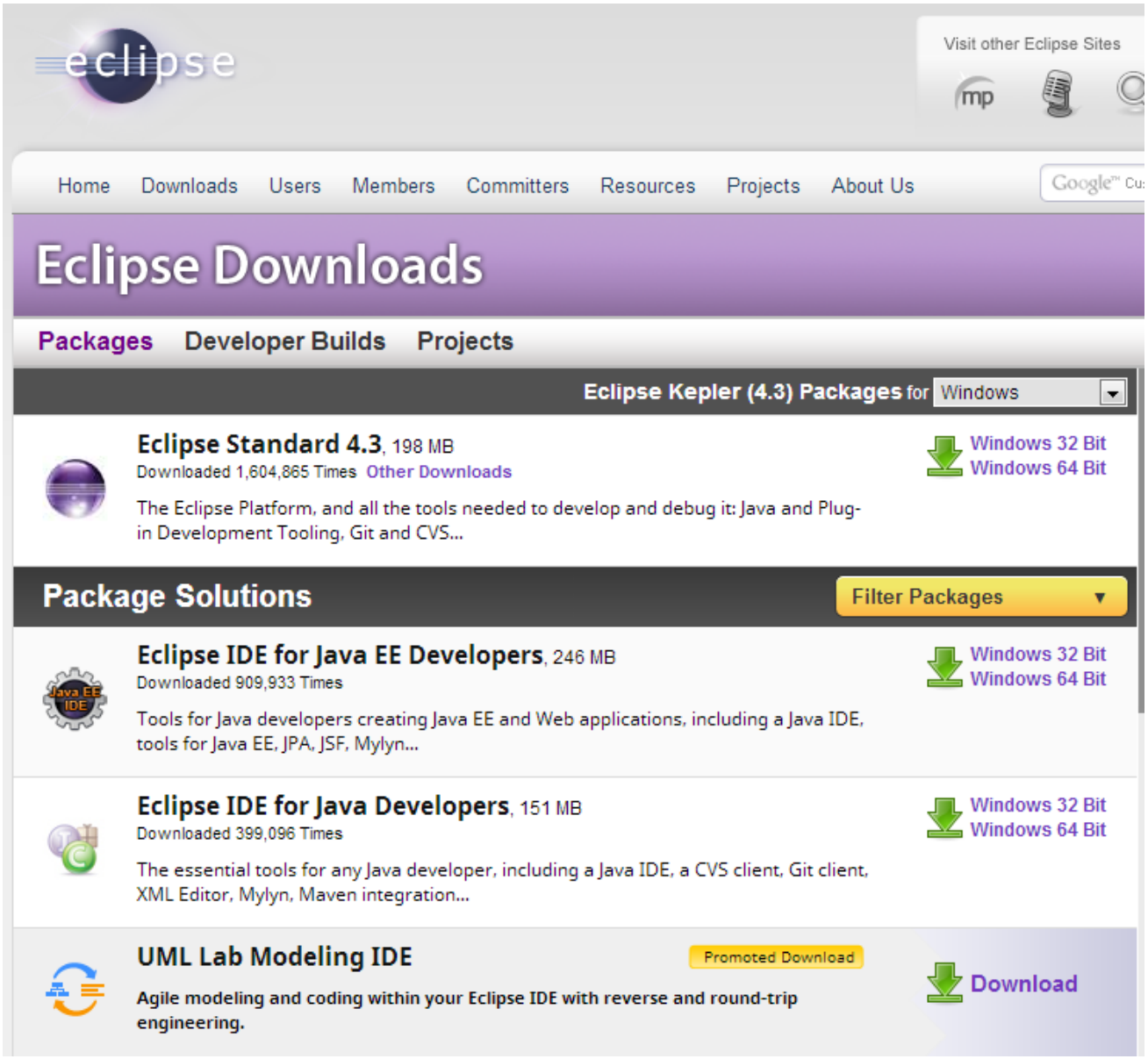
If you install JDK 1.5 version or later, you do not need to set an environment variable (JAVA_HOME) since you can access it by copying an execution file to the system path.

Installing Eclipse

You can use Eclipse without installing it by simply downloading and unzipping it.

Access the following link and download the Eclipse Standard version, or choose a file appropriate for your development environment.

http://www.eclipse.org/downloads/

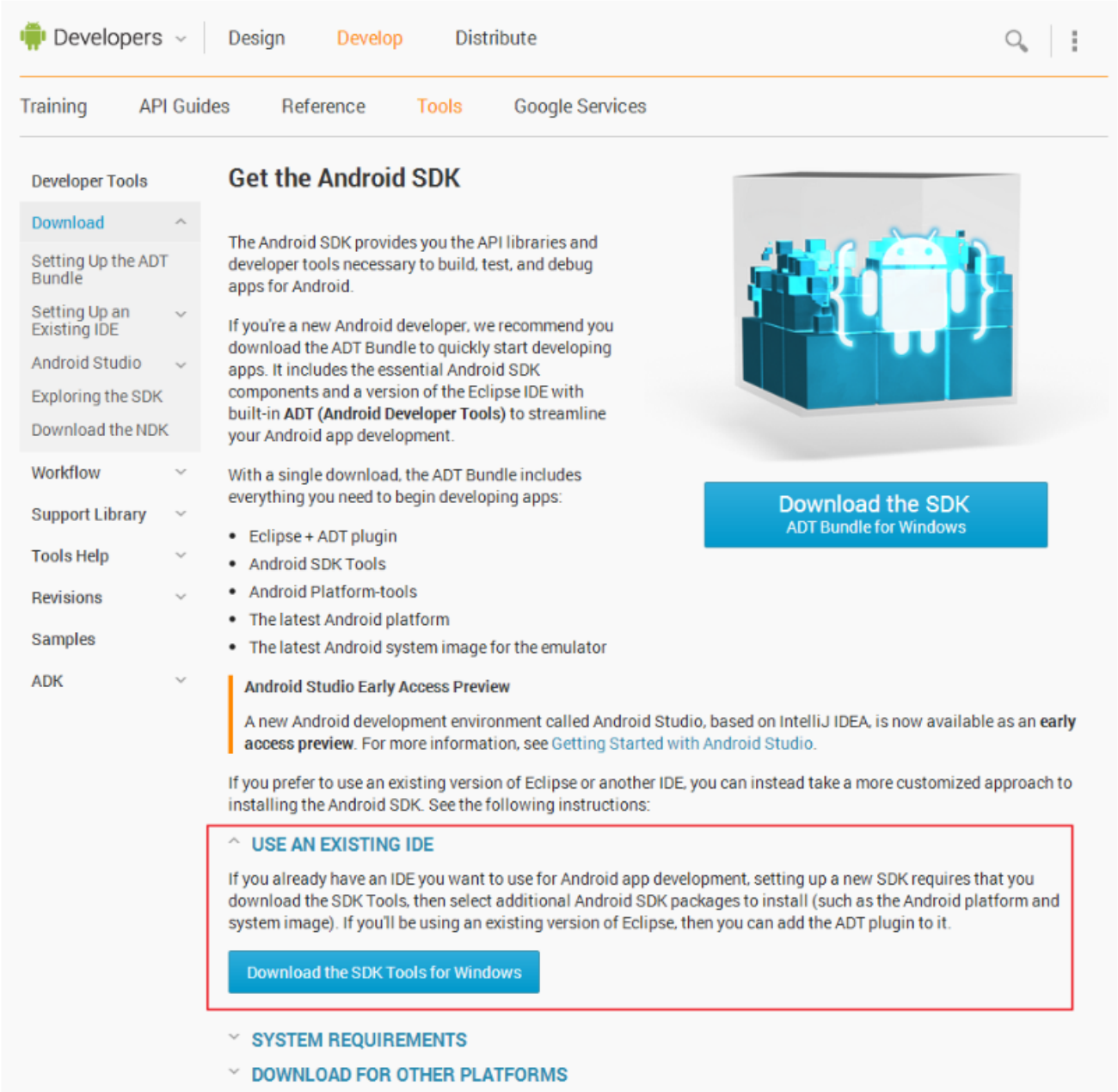

eclipse
Visit other Eclipse Sites
Home
Downloads
Users
Members
Committers
Resources
Projects
About Us
Eclipse Downloads
Packages
Developer Builds
Projects
Eclipse Kepler (4.3) Packages for Windows
Eclipse Standard 4.3, 198 MB
Downloaded 1,604,865 Times Other Downloads
The Eclipse Platform, and all the tools needed to develop and debug it: Java and Plug-in Development Tooling, Git and CVS...
Windows 32 Bit
Windows 64 Bit
Package Solutions
Filter Packages
Eclipse IDE for Java EE Developers, 246 MB
Downloaded 909,933 Times
Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...
Windows 32 Bit
Windows 64 Bit
Eclipse IDE for Java Developers, 151 MB
Downloaded 399,096 Times
The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven integration...
Windows 32 Bit
Windows 64 Bit
UML Lab Modeling IDE
Promoted Download
Agile modeling and coding within your Eclipse IDE with reverse and round-trip engineering.
Download

Installing Android SDK

For Android projects, you need to install the Android SDK for related API. The relevant file can be downloaded from the Android developer site.

Download the ADT Bundle package which includes Eclipse with the ADT plug-in, or download an SDK installation file separately. To download the SDK installation file separately, visit the site below, select the **USE AN EXISTING IDE** link from the menu toward the bottom of the page, and click **Download SDK Tools for Windows**.

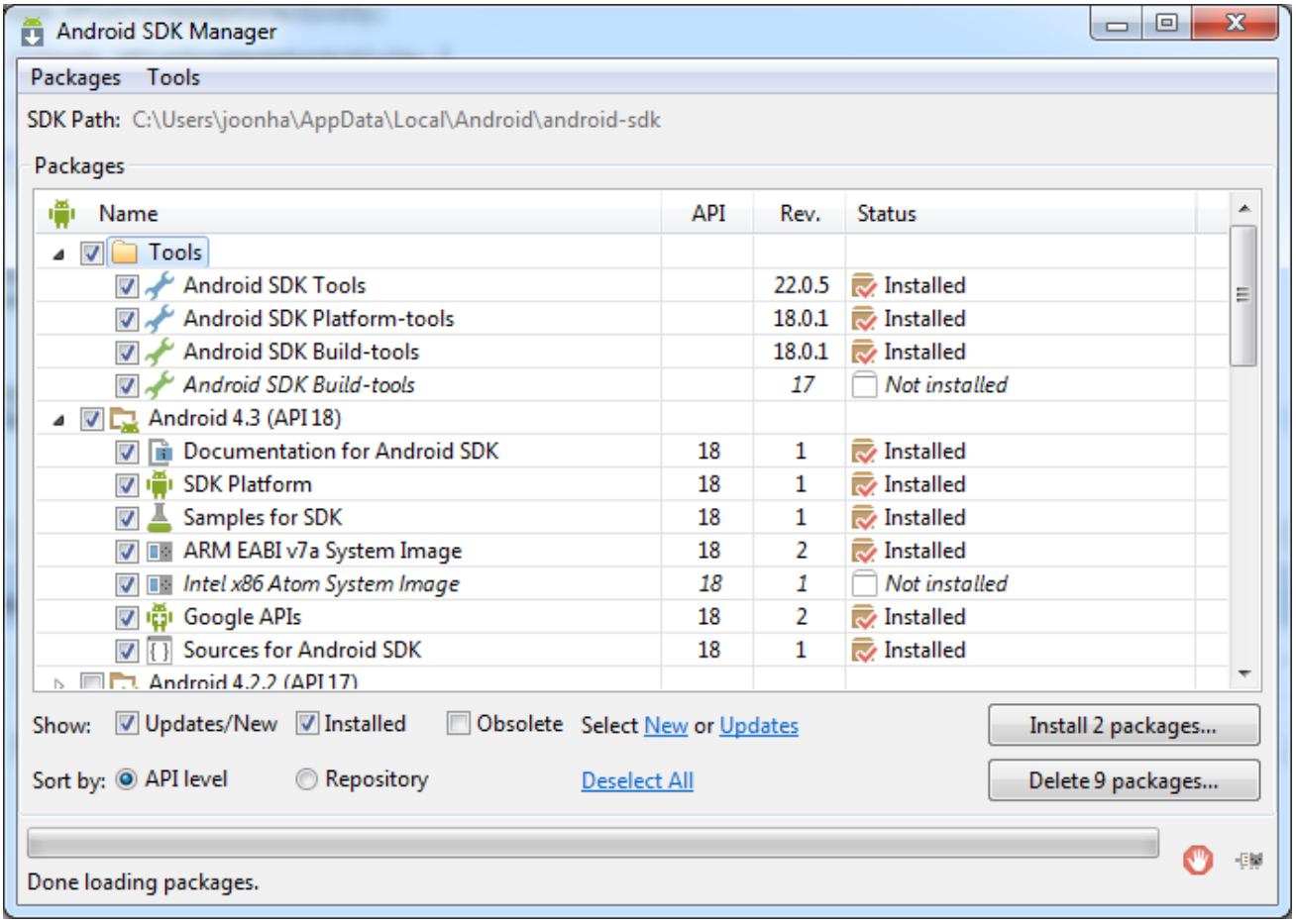
http://developer.android.com/sdk/index.html


Developers
Design
Develop
Distribute
Training
API Guides
Reference
Tools
Google Services
Developer Tools
Download
Setting Up the ADT Bundle
Setting Up an Existing IDE
Android Studio
Exploring the SDK
Download the NDK
Workflow
Support Library
Tools Help
Revisions
Samples
ADK
Get the Android SDK
The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.
If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in ADT (Android Developer Tools) to streamline your Android app development.
With a single download, the ADT Bundle includes everything you need to begin developing apps:
Eclipse + ADT plugin
Android SDK Tools
Android Platform-tools
The latest Android platform
The latest Android system image for the emulator
Download the SDK
ADT Bundle for Windows
Android Studio Early Access Preview
A new Android development environment called Android Studio, based on IntelliJ IDEA, is now available as an early access preview. For more information, see Getting Started with Android Studio.
If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions:
USE AN EXISTING IDE
If you already have an IDE you want to use for Android app development, setting up a new SDK requires that you download the SDK Tools, then select additional Android SDK packages to install (such as the Android platform and system image). If you'll be using an existing version of Eclipse, then you can add the ADT plugin to it.
Download the SDK Tools for Windows
SYSTEM REQUIREMENTS
DOWNLOAD FOR OTHER PLATFORMS

Run the downloaded installation file.

When the installation is completed, the SDK Manager will run. SDK Manager provides Android APIs required for the development as well as functions to download related tools. Select **Tools** and **Extras**, and install the required Android APIs.

If later you need to install additional APIs, you can run SDK Manager again.

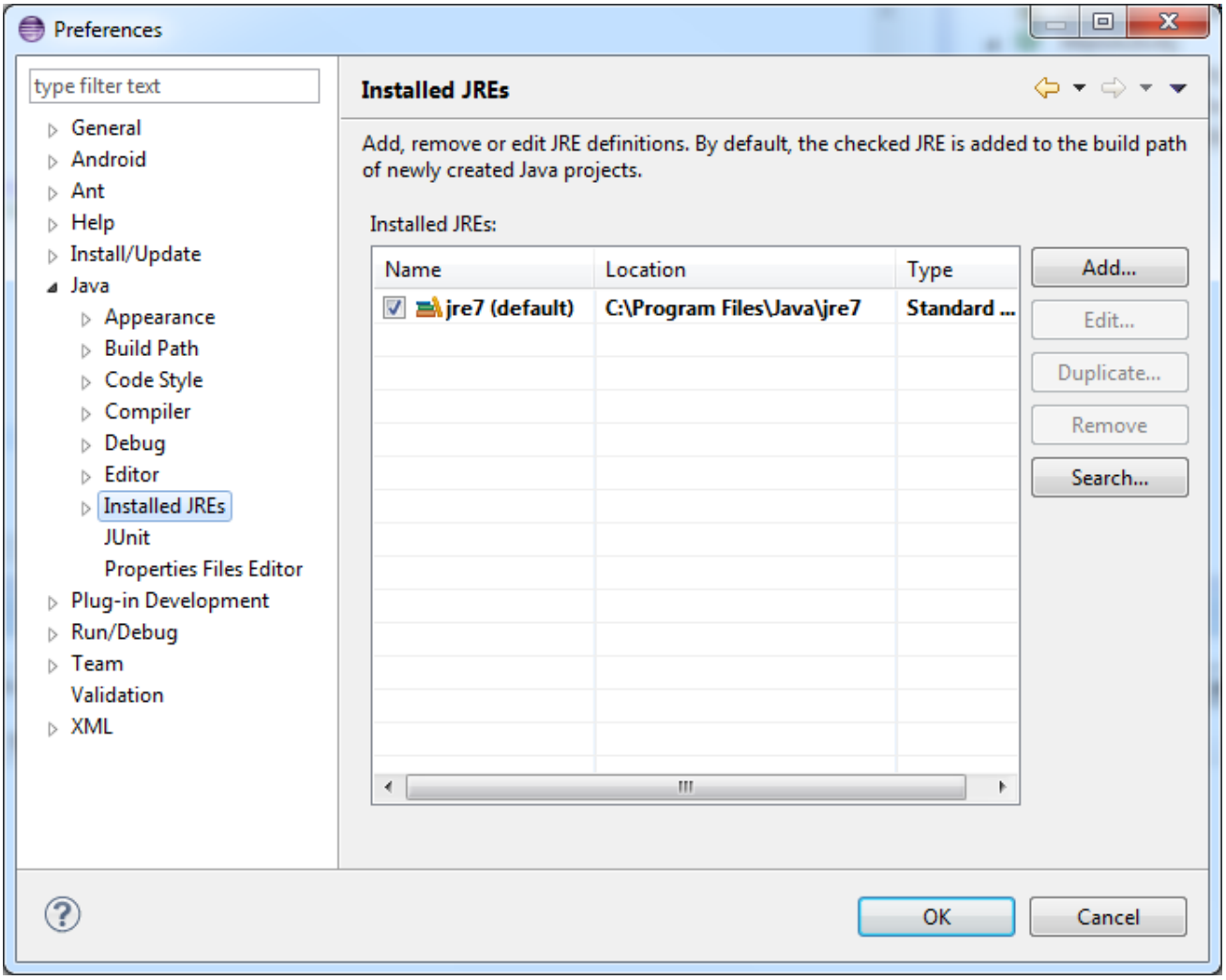

Android SDK Manager
Packages Tools
SDK Path: C:\Users\joonha\AppData\Local\Android\android-sdk
Packages
Name API Rev. Status
Tools
Android SDK Tools 22.0.5 Installed
Android SDK Platform-tools 18.0.1 Installed
Android SDK Build-tools 18.0.1 Installed
Android SDK Build-tools 17 Not installed
Android 4.3 (API 18)
Documentation for Android SDK 18 1 Installed
SDK Platform 18 1 Installed
Samples for SDK 18 1 Installed
ARM EABI v7a System Image 18 2 Installed
Intel x86 Atom System Image 18 1 Not installed
Google APIs 18 2 Installed
Sources for Android SDK 18 1 Installed
Show: Updates/New Installed Obsolete Select New or Updates
Install 2 packages...
Sort by: API level Repository
Deselect All
Delete 9 packages...
Done loading packages.

It will take a while to install Android APIs depending on your network environment.

ADT Plug-in Installation

After installing Eclipse, please check if the JRE (Java Runtime Environment) version corresponds to the installed JDK version. You can set the JRE using the following menu.

```
Window > Preferences > Java > Installed JREs
```



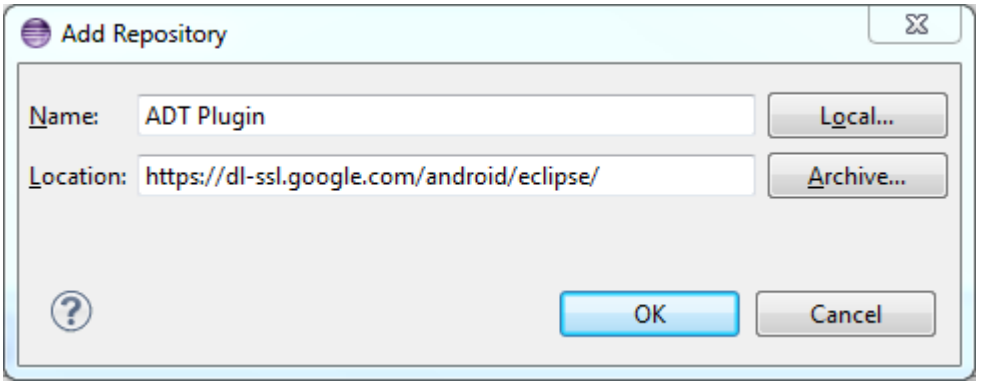
To construct Android projects, you need to install the ADT plug-in. Add developer tools using the following menu.

```
Help > Install New Software
```

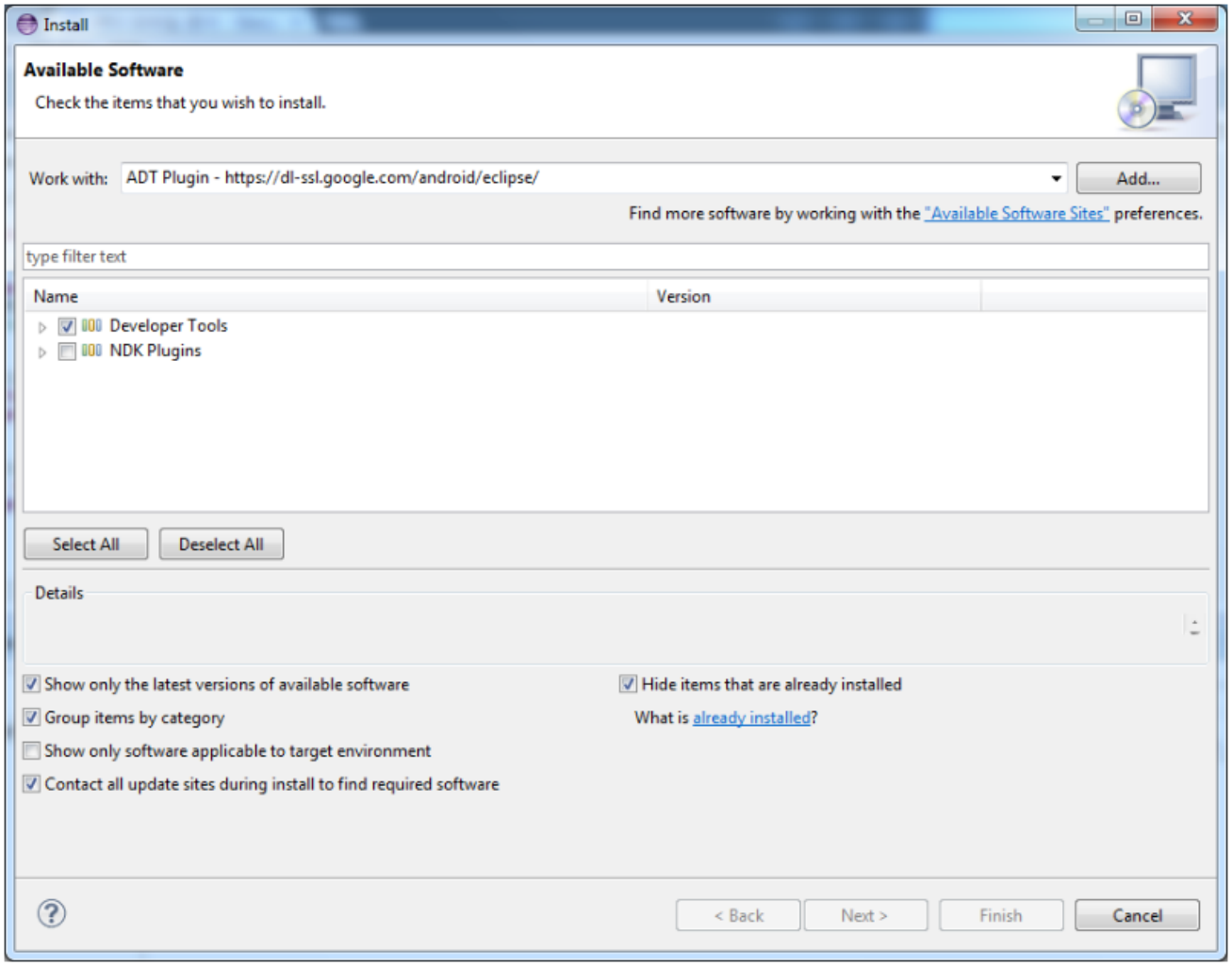
> The menu for installing additional functions can vary according to the Eclipse versions in use. For pre-Eclipse 3.4 versions, use `Help` > `Software Updates`.
> This chapter illustrates use of the Eclipse 4.3(Kepler) version.

Click the **Add** button next to the *Work with* item, and enter the Repository information as follows.

```
Name: ADT Plugin
Location: https://dl-ssl.google.com/android/eclipse/
```



To start the installation, select the **Developer Tools** checkbox among the available items.



After the installation is completed, Eclipse will be restarted.

8.2 App Project Development

Once the application development environment is configured, you can develop apps running on the Android OS using Eclipse.

Before you start app projects, archive files generated by applications developed on Nexacro Studio will be located in the designated path.

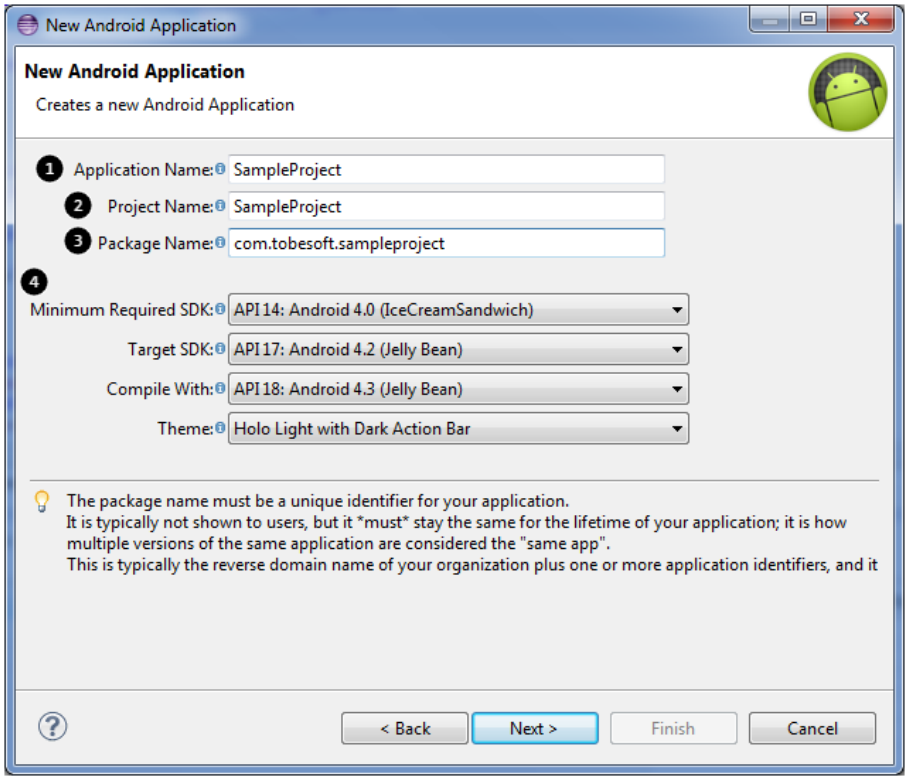
Creating Projects

You need to create Android projects which include applications developed by Nexacro Platform and to set the basic environment. You can create a new project using the menu below.

```
File > New > Android Application Project
```

If you can't see the *Android Application Project item*, the ADT plug-in was not installed successfully. Check the installation of ADT plug-in again.

Project settings can differ based on the version of Eclipse or Android SDK.


New Android Application
New Android Application
Creates a new Android Application
1 Application Name: SampleProject
2 Project Name: SampleProject
3 Package Name: com.tobesoft.sampleproject
4
Minimum Required SDK: API 14: Android 4.0 (IceCreamSandwich)
Target SDK: API 17: Android 4.2 (Jelly Bean)
Compile With: API 18: Android 4.3 (Jelly Bean)
Theme: Holo Light with Dark Action Bar
The package name must be a unique identifier for your application.
It is typically not shown to users, but it *must* stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app".
This is typically the reverse domain name of your organization plus one or more application identifiers, and it
< Back
Next >
Finish
Cancel

Callout	Item	Description
❶	Application Name	It requires a name of an app to be created.
❷	Project Name	It requires a name of a project to be created. The project name shall be a sole name in the workspace in use.
❸	Package Name	It requires a domain-type name of a package
❹	Minimum Required SDK	Nexacro Platform supports Android 2.3(Gingerbread) or later.

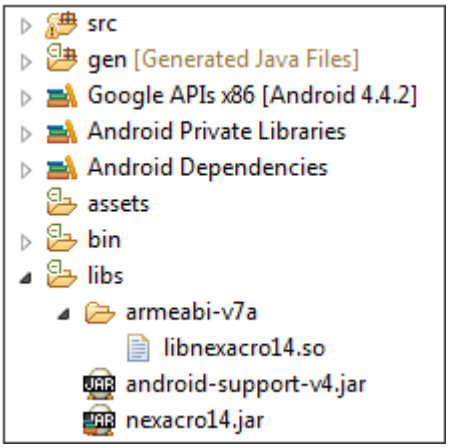
For Android 4.4 (Kitkat) or later SDK the appcompat_v7 project is automatically created for downward compatibility if the *Minimum Required SDK* is set below 3.2 (Honeycomb). Refer to http://developer.android.com/tools/support-library/features.html.
However, this setting does not have any effect when you create Nexacro Platform applications. Therefore, it is not recommended to create the project. To deal with this situation, you can set the *Minimum required SDK* to **4.0(IceCreamSandwich)** and create a project. You can then designate a downward version as you want when setting *AndroidManifest.xml*.

Nexacro Platform Library Settings

To optimize the Android project environment for XPLATFORM, you need to configure additional Nexacro Platform libraries:

1. Copy the provided library files to the *libs* folder subordinate to the project folder.
2. Copy the *nexacro14.jar* file subordinate to the *libs* folder.
3. Create a new folder subordinate to the libs *folder* and name it *armeabi-v7a*, and
4. Copy the *libnexacro14.so* file into the new folder

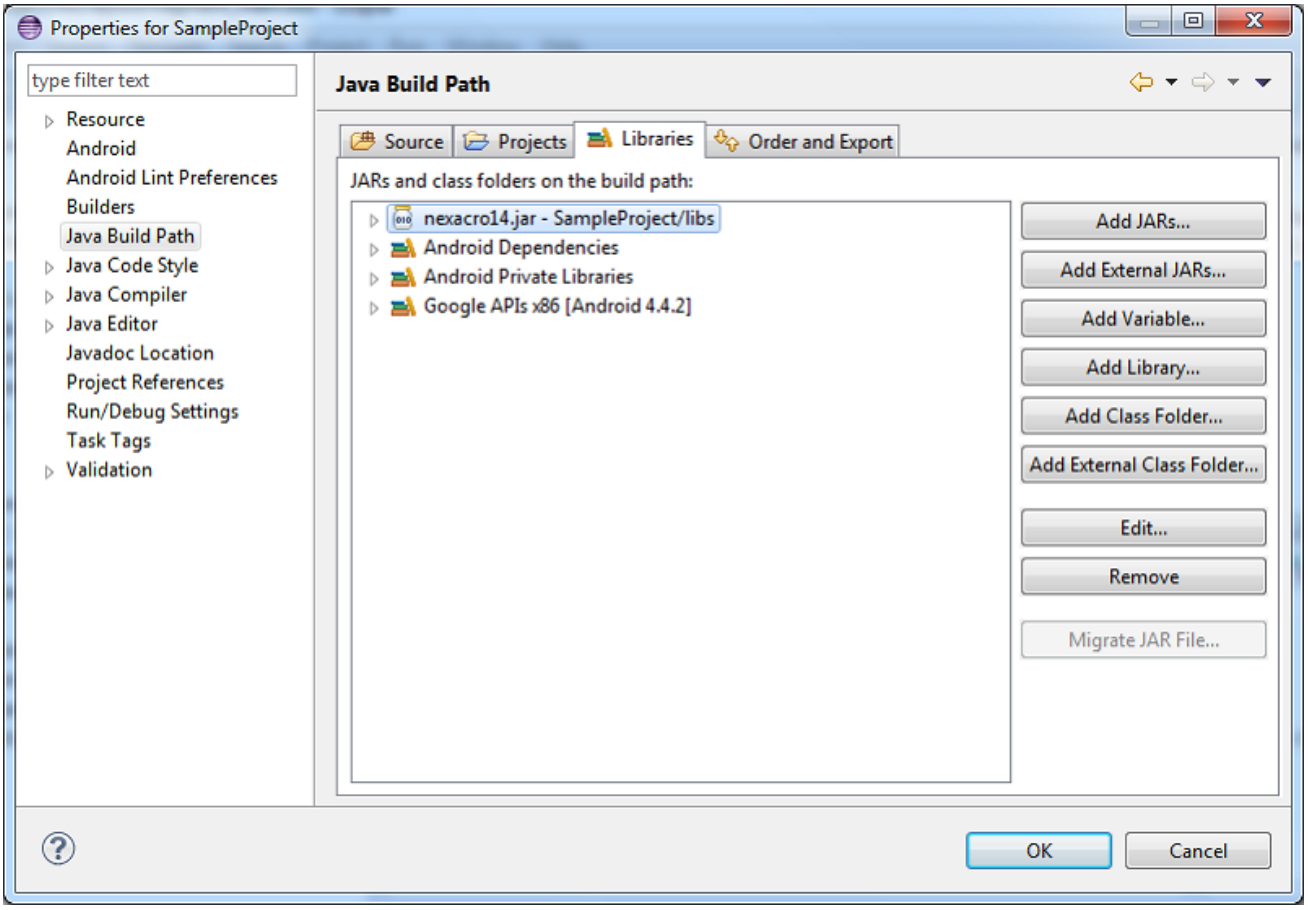
The folder structure should now resemble the following.



The *libnexacro14.so* file is a library developed in the C/C++ language. The "abi" in the folder name is an abbreviation for "Application Binary Interface".

Set *Java Build Path* to refer to the added Nexacro Platform libraries on the project:

1. Access **File > Properties > Java Build Path**.
2. Select the **Libraries** tab.
3. Click the **Add JARs** button.
4. Select the *nexacro14.jar* file in the libs folder.



Resource Settings

Configure the images, icons, messages and layouts that will be used by the project applications. The folder structure will resemble the following.

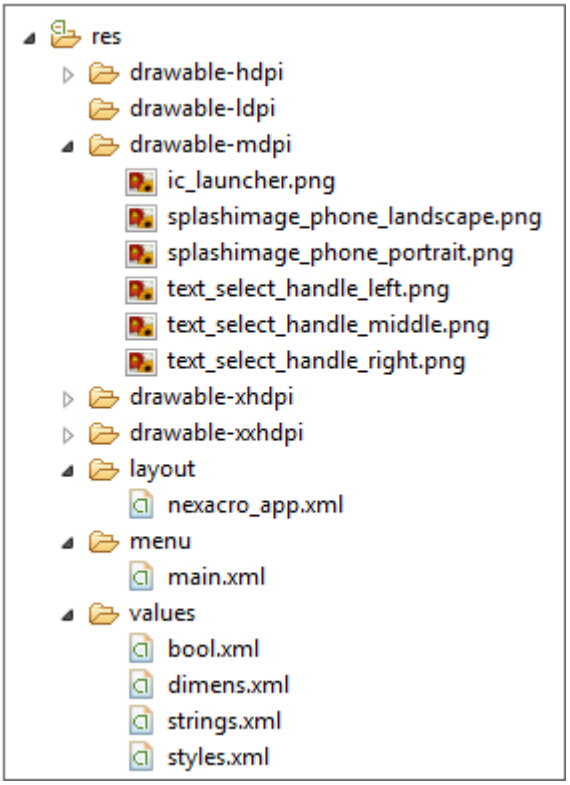


Image Settings

Designate the loading images and app icons in the *drawable-mdpi* folder subordinate to the *res* folder.

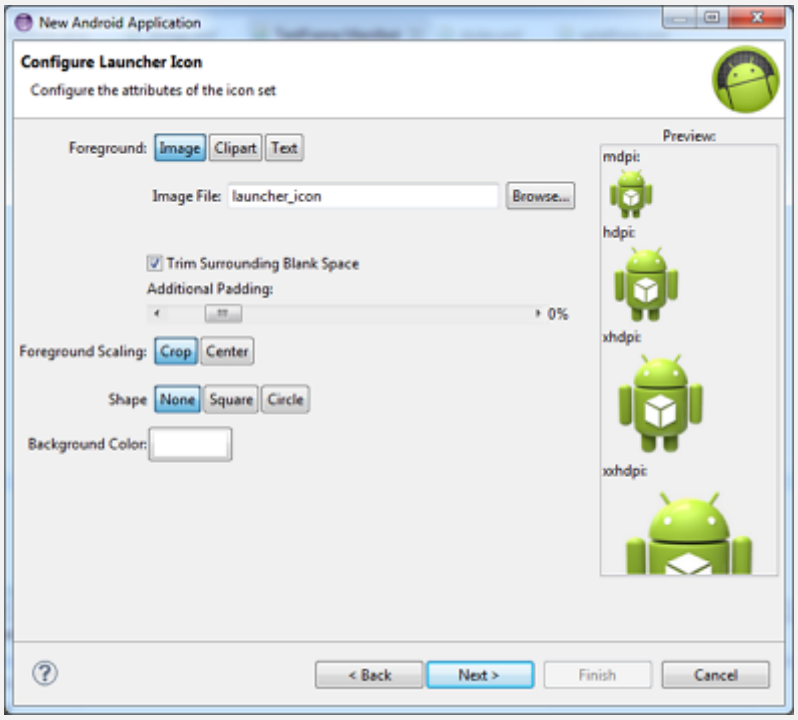
Applied image files can be adjusted to the resolution supported by apps. Please refer to the Android manual for details.

http://developer.android.com/guide/practices/screens_support.html

Item	Description
icon.png	Icon image installed in the backround of an application The designated file name can be changed in the AndroidManifest.xml file
splashimage_phone_landscape.png	Horizontal splash image for the Android phone (all lowercase)
splashimage_phone_portrait.png	Vertical splash image for the Android phone (all lowercase)
splashimage_pad_landscape.png	Horizontal splash image for the Android tablet (all lowercase)
splashimage_pad_portrait.png	Vertical splash image for the Android tablet (all lowercase)

A file name that starts with *text_select_handle~* corresponds to an image to be used to handle a specific area when selecting text. In the current version, you need to designate the relevant image.

An icon file can be configured in detail in the *Configure Launcher Icon* page when creating a project.



Message Settings

Configure settings for messages used when running apps. Revise the *strings.xml* file, generated when the project was created, located in the *values* folder subordinate to the *res* folder.

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Nexacro 14</string>
    <string name="action_settings">Settings</string>

    <!-- Clipboard -->
    <string name="copy">Copy</string>
    <string name="paste">Paste</string>
    <string name="cut">Cut</string>
    <string name="close">Close</string>
    <string name="selectall">Select All</string>
    <string name="selectword">Select Word</string>

    <string name="needupdate">It is need to update. \r\nAfter completing,
```

```
Please restart.</string>
    <string name="loadingFail">First loading is fail. \r\nPlease
restart.</string>
    <string name="updateFail">Update is fail. \r\nPlease restart.</string>
    <string name="notexist">Start file is NOT exist. \r\nPlease
restart.</string>
    <string name="BeingUpdated">Being updated.</string>
    <string name="wantreplace">There is a file of the save name. Do you want to
replace? </string>
    <string name="ok">OK</string>
    <string name="cancel">Cancel</string>
    <string name="move">Move</string>
    <string name="upper">Upper</string>
    <string name="filter">Filter</string>
    <string name="home">Home</string>
    <string name="nofilename">No File Name.</string>
    <string name="checkforupdates">Check for updates.</string>
    <string name="installforupdates">Install for updates.</string>
    <string name="downloadingforupdates">Downloading for updates.</string>
    <string name="force_close">Update is Completed.</string>
    <string name="LoadingPleaseWait">Loading&amp;#8230; \r\nPlease
Wait&amp;#8230;</string>

    <!-- Accessibility widget role -->
    <string name="alert">alert</string>
    <string name="pushbutton">pushbutton</string>
    <string name="combobox">combobox</string>
    <string name="checkbutton">checkbutton</string>
    <string name="radiobutton">radiobutton</string>
    <string name="text">text</string>
    <string name="list">list</string>
    <string name="listitem">listitem</string>
    <string name="statictext"></string>
    <string name="graphic">image</string>
    <string name="spacebar">space bar</string>
    <string name="textdeleted">deleted</string>

    <!-- Accessibility widget statue -->
    <string name="checked">checked</string>
    <string name="unchecked">unchecked</string>
```

```
    <!-- Accessibility reaction -->
    <string name="click">click</string>

    <!-- Splash message  -->
    <string name="start_initialize">Start Initialize</string>
    <string name="check_license">Check License</string>
    <string name="load_frameworkmodule">Load Framework Module</string>
    <string name="load_extendmodule">Load Extend Module</string>
    <string name="execute_frameworkscript">Execute Framework Script</string>
    <string name="load_application">Load Application</string>
    <string name="load_failapplication">Load Fail Application Initialize. Click
And Close</string>

</resources>
```

If you want to add messages in other languages, such as Korean or Japanese, add corresponding folders, such as *values-ko* and *values-ja*, subordinate to the *res* folder, and add a *strings.xml* file to each.

The `<string name="app_name">` value defines the icon name actually installed on the device. Change the icon name by revising this value.

Add a bool.xml file.

bool.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <bool name="is_jellybean_or_higher">false</bool>
</resources>
```

Layout Settings

Define the display layouts of apps on devices. Set the layout as *default* since it is for covering Nexacro Platform applications.

Create *nexacro_app.xml* in the *layout* folder subordinate to the *res* folder as listed here.

nexacro_app.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- nexacro_app.xml -->
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:id="@+id/nexacro_activity"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/transparent"
    tools:context=".deviceAPI.FileDialogActivity" >

    <com.nexacro.view.NexacroLayout
        android:id="@+id/nexacro_layout"
        android:background="@android:color/transparent"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </com.nexacro.view.NexacroLayout>

</FrameLayout>
```

Build Environment Settings

Make adjustments to the *MainActivity.java* and *AndroidManifest.xml* files as described in this section. These files were created when you added Android projects to the Nexacro Platform environment.

MainActivity.java

When you create a project, you will designate an activity name. It shall be created as *MainActivity.java* unless you change the name. Select the *MainActivity.java* file, which resides in the *src* folder subordinate to the designated package, and revise it as follows.

In the file, the highlighted `setBootstrapURL` parameter defines a server URL where the *start_android.json* file created on Nexacro Studio is distributed. And a parameter included in the `setProjectURL` method defines the path up to /.

MainActivity.java

```
package com.tobesoft.sampleproject;

import com.nexacro.NexacroUpdatorActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends NexacroUpdatorActivity {
```

```
    public MainActivity() {
        super();
        setBootstrapURL("http://172.10.10.10:8080/nexacro/start_android.json");
        setProjectURL("http://172.10.10.10:8080/nexacro/");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Intent intent = getIntent();
        if(intent != null) {
            String bootstrapURL = intent.getStringExtra("bootstrapURL");
            String projectUrl = intent.getStringExtra("projectUrl");
            if(bootstrapURL != null) {
                setBootstrapURL(bootstrapURL);
                setProjectURL(projectUrl);
            }
        }

        super.onCreate(savedInstanceState);
    }

    @Override
    public void setContentView(View view) {
        super.setContentView(view);
    }
}
```

If you want to access the server directly without creating a project archive file, add `setStartupURL` method designating *xadl.js* as below. At this point, only the *engine* item shall be set to start_android.jsonfile.

```
super();
setBootstrapURL("http://172.10.10.10:8080/nexacro/start_android.json");
setStartupURL
("http://172.10.10.10:8080/nexacro/HBDeviceTest/HBDeviceTest.xadl.js");
```

AndroidManifest.xml

The *AndroidManifest.xml* file is located in the project root, and includes the basic application information as well as the information required to execute certain features. Revise the file as illustrated below. The information about the highlighted package and `MainActivity` should be designated according to the project settings.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tobesoft.sampleproject"
    android:installLocation="auto"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="19" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

<!--This portion can be deleted if you use the asset/archive feature.
Otherwise, you must not delete this portion-->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:hasCode="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.tobesoft.sampleproject.MainActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:theme="@android:style/Theme.NoTitleBar" >
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name="com.nexacro.NexacroActivity"
            android:alwaysRetainTaskState="true"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:launchMode="singleTop"
            android:theme="@android:style/Theme.NoTitleBar"
            android:windowSoftInputMode="adjustResize" >
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="google_play_services_version" />
    </application>
</manifest>
```

Depending on the features of an application, you need to designate additional information such as permission. For instance, you will be required to insert the below code if you intend to use the camera feature.

```
...
<uses-permission android:name="android.permission.CAMERA" />
<activity android:name="com.nexacro.deviceAPI.CameraListener"
    android:screenOrientation="landscape">
    <intent-filter>
           <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus"
    android:required="false"/>
...
```

The following sections present you with available features and their syntax rules. Copy and paste necessary items into the *AndroidManifest.xml* file.

DeviceAPI

```
<!-- AudioPlayer, AudioRecorder -->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />

<!-- Camera -->
<uses-permission android:name="android.permission.CAMERA" />
<activity android:name="com.nexacro.deviceAPI.CameraListener"
android:screenOrientation="landscape">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus"
android:required="false"/>

<!-- Contact -->
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />

<!-- ExternalAPI -->
<uses-permission android:name="android.permission.GET_TASKS" />

<!-- Geolocation, Map -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />

<!-- Map -->
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-library android:name="com.google.android.maps" />
<meta-data android:name="com.google.android.maps.v2.API_KEY"
android:value="user value" />

<!-- Notification -->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="com.nexacro.permission.C2D_MESSAGE"
android:protectionLevel="signature" />
<uses-permission android:name="com.nexacro.permission.C2D_MESSAGE" />
```

```
<receiver
    android:name="com.nexacro.notification.NexacroNotificationReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.nexacro" />
    </intent-filter>
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.nexacro" />
    </intent-filter>
</receiver>
<service android:name="com.nexacro.notification.NexacroNotificationService" />

<!-- SMS -->
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />

<receiver android:name="com.nexacro.deviceAPI.SmsRecv">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.provider.Telephony.WAP_PUSH_RECEIVED" />
        <data android:mimeType="application/vnd.wap.mms-message" />
    </intent-filter>
</receiver>"

<!-- Vibrator -->
<uses-permission android:name="android.permission.VIBRATE" />

<!-- FileDialog -->
<activity android:name="com.nexacro.deviceAPI.FileDialogActivity"
android:screenOrientation="sensor">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

To use Google maps, you need to get an API KEY and enter the key. Refer to the link below for details.

https://developers.google.com/maps/documentation/android/start#obtain_a_google_maps_api_key

GlobalAPI

```
<!-- Accounts -->
<uses-permission android:name="android.permission.GET_ACCOUNTS" />

<!-- Phone numbers -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

<!-- Phone calling -->
<uses-permission android:name="android.permission.CALL_PHONE" />

<!-- Device names -->
<uses-permission android:name="android.permission.BLUETOOTH"/>
```

Asset/archive feature

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

Cache feature

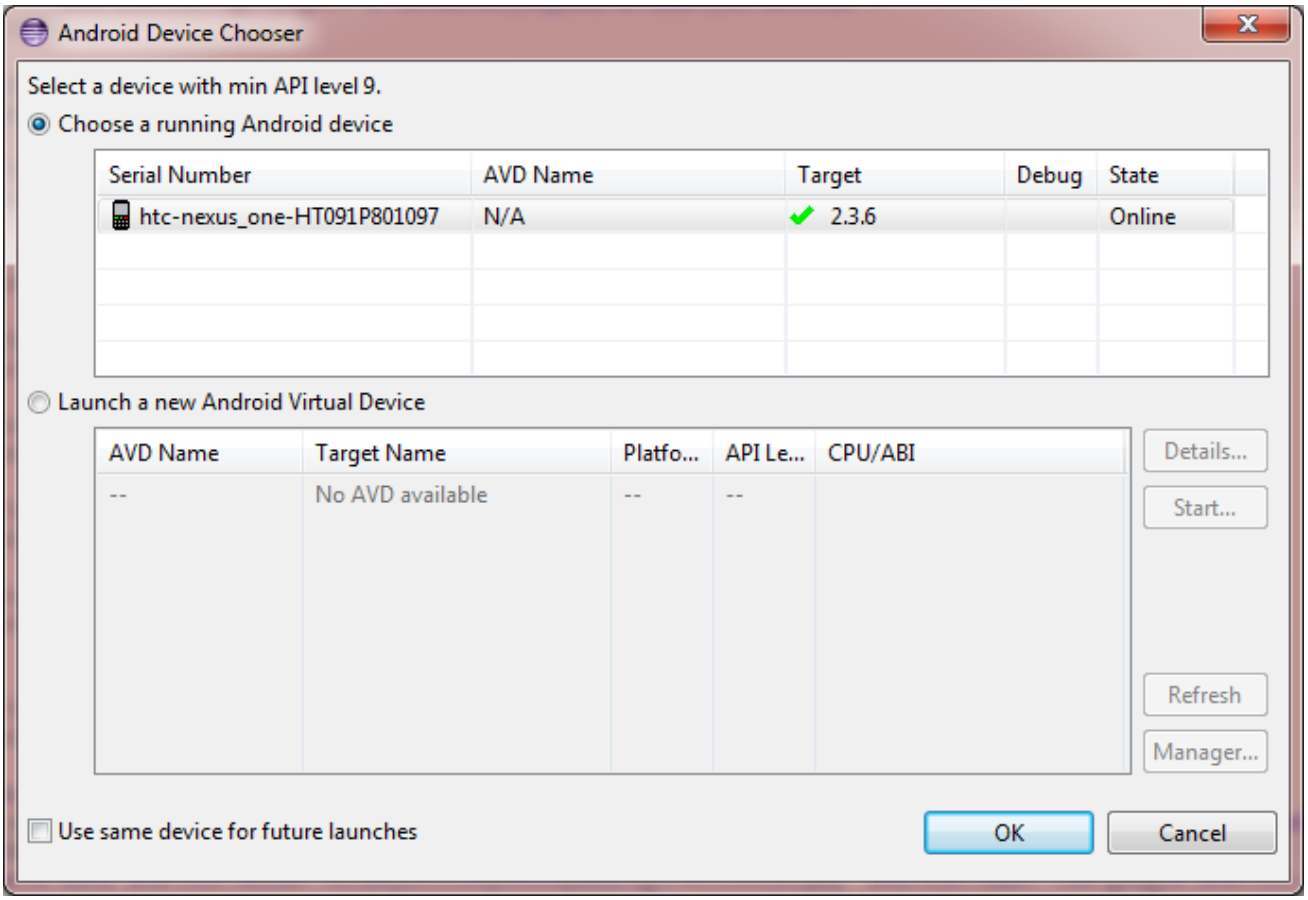
```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_INTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE" />
```

8.3 App Test

Test your applications directly by connecting an Android device to a PC using USB cable. When you connect the device directly, you will need to set the **Application > Development > USB debugging** item, which is an application development option, and also install a driver appropriate for the device.

You will use the **Run** menu on Eclipse to create an APK and install/run the APK on the device:

1. Choose the device from the Android Device Chooser page.



2. Select the **RUN** action from the Run menu.

For a Nexus One or Optimus One, if there is insufficient memory, the application will not be installed normally unless the user adds an external storage drive.

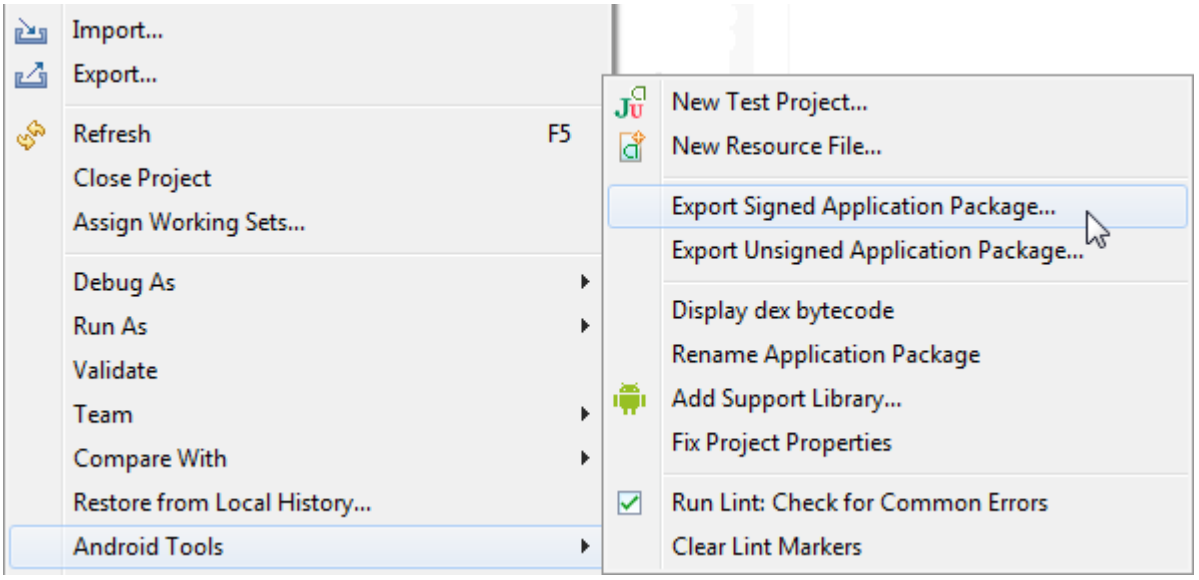
Android does not support testing by using a virtual device.

For creating keystores and signed APK, refer to the Google webpage.

8.4 Creating Installation Files

To distribute completed apps, you need to create keystores which acknowledge apps and signed APK files. Keystore files will be created once for the first time, and the initial keystore files will be used for updates.

Choose the project and select the **Android Tools > Export Signed Application Package** item on the context menu.



Refer to the Google webpage for creating keystores and signed APK.

http://developer.android.com/guide/publishing/app-signing.html

9.
App Distribution

As app distribution is subject to market operators, this chapter covers the outline of app distribution. For details, refer to the web pages described in each section.

9.1 How to Distribute Archive Files

You need to copy archive files created on Nexacro Studio to a distribution server. Nexacro Studio does not support copying to the distribution server. For that reason, you have to copy those files directly to the server yourself.

The file to be copied are located in the same path as *startinfo_android[ios].json*, which is specified in each app project for each device.

The table below is a list of archive files to be distributed to the distribution server.

Distribution List	File Name	Contents
ArchiveInfo	startinfo_android[ios].json	Path information of archive files and base libraries
Base Library	nexacro14lib.zip	Zipped-type Nexacro Platform frameworks
Project Archive File	Archive00.xzip	Nexacro Platform HTML5 project files developed on Nexacro Studio
Theme Archive File	[ThemeName].zip	Nexacro Platform HTML5 project themes
Run Archive File	Run.zip	Component information and initially executed ADL JavaScript file path information (applied to iOS operating system only)

9.2 How to Distribute Apps through Android Markets

Google Play

To distribute apps through Google Play, a Google Play account is needed.

https://play.google.com/apps/publish/signup

You can distribute apps on Google Play by registering created APK files on Google Play according to a guide provided on the Google Play webpage. No inspection period is needed. Apps can be distributed right away as long as APK files do not have any problems.

SamsungApps

To distribute apps through SamsungApps, you need to enroll in SamsungApps as a seller.

http://seller.samsungapps.com

After creating a seller account, you can register APK files according to a guide on the webpage. The apps will be distributed after going through an inspection process. SamaungApps has its own inspection process, and as long as the files do not violate any items of the inspection list on the webpage, the apps will be distributed after the inspection is completed.

T Store

To distribute apps through T Store, you need to enroll in T Store's developer center.

http://dev.tstore.co.kr

After creating an account, you can register APK files according to a guide on the webpage. The apps will be distributed after going through an inspection process. T Store has its own inspection process, and as long as the files do not violate any items of the inspection list on the webpage, the apps will be distributed after the

inspection is completed.

Olleh Market

To distribute apps through Olleh Market, you need to enroll in Olleh Market's seller support center.

http://seller.ollehmarket.com

After creating an account, you can register APK files according to a guide on the webpage. The apps will be distributed after going through an inspection process. Olleh Market has its own inspection process, and as long as the files do not violate any items of the inspection list on the webpage, the apps will be distributed after the inspection is completed.

U+ App Market

To distribute APK files through U+ App Market, you need to enroll in U+ App Market's developer center.

http://devpartner.lguplus.co.kr/

After creating an account, you can register APK files according to a guide on the webpage. The apps will be distributed after going through an inspection process. U+ App Market has its own inspection process, and as long as the files do not violate any items of the inspection list on the webpage, the apps will be distributed after the inspection is completed.

9.3 How to Distribute Android APK Files through Private Servers

Android APK files can be distributed using private servers or email. (Android APK files can be distributed anywhere as long as those files can be downloaded and installed on Android devices.)

Android devices are initially set to block installation of apps not distributed through markets. For that reason, if you want to distribute apps using private servers, check the checkbox on the **Settings > Application > Unknown source** menu.

Setting menu items can differ depending on device manufacturers. Please refer to Android developer center or each device manual for details.
http://developer.android.com/guide/publishing/publishing_overview.html

Distribution using a Web Server

You need to upload an APK file on your web browser so that you can obtain an URL for the relevant APK. The app can be downloaded and installed on a device by entering the URL on the Android device's browser.

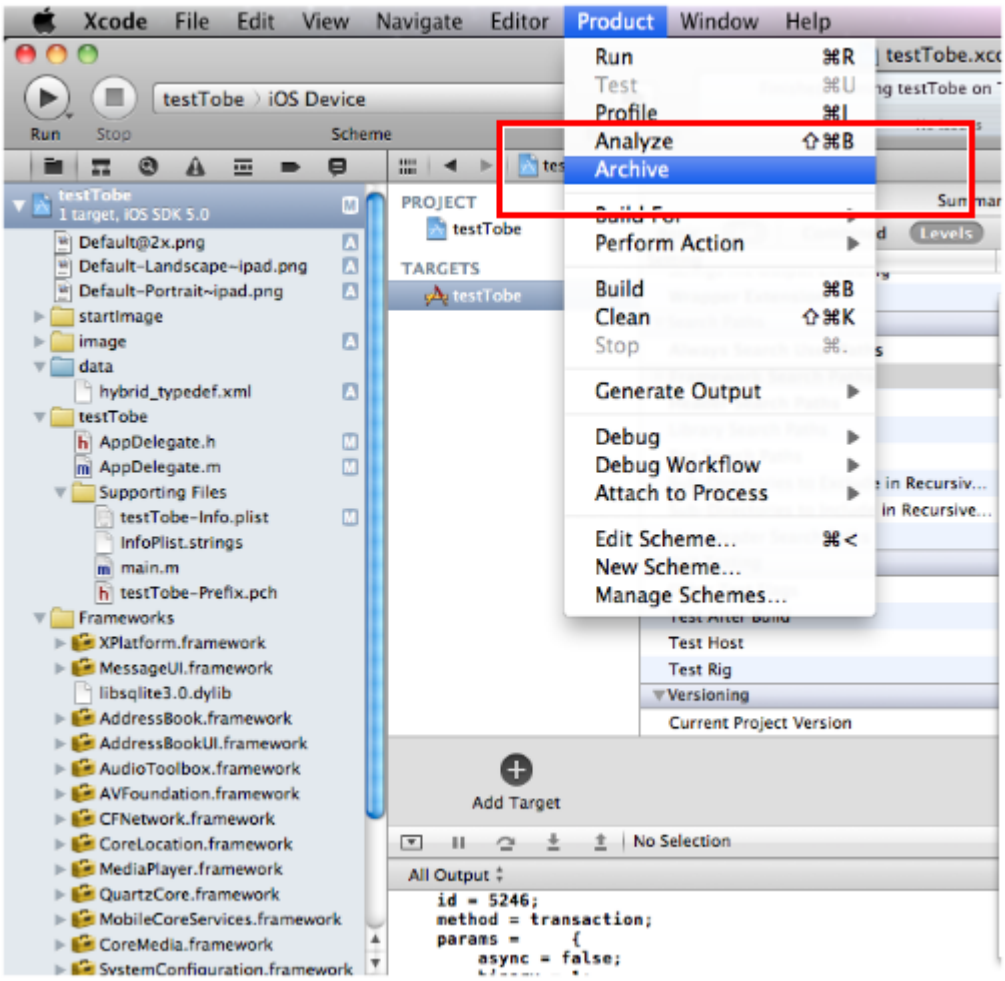
Distribution using Email

You need to attach an APK file to an email. The app can be downloaded and installed by opening the email containing the APK file on the Android device.

9.4 How to Distribute iOS Apps

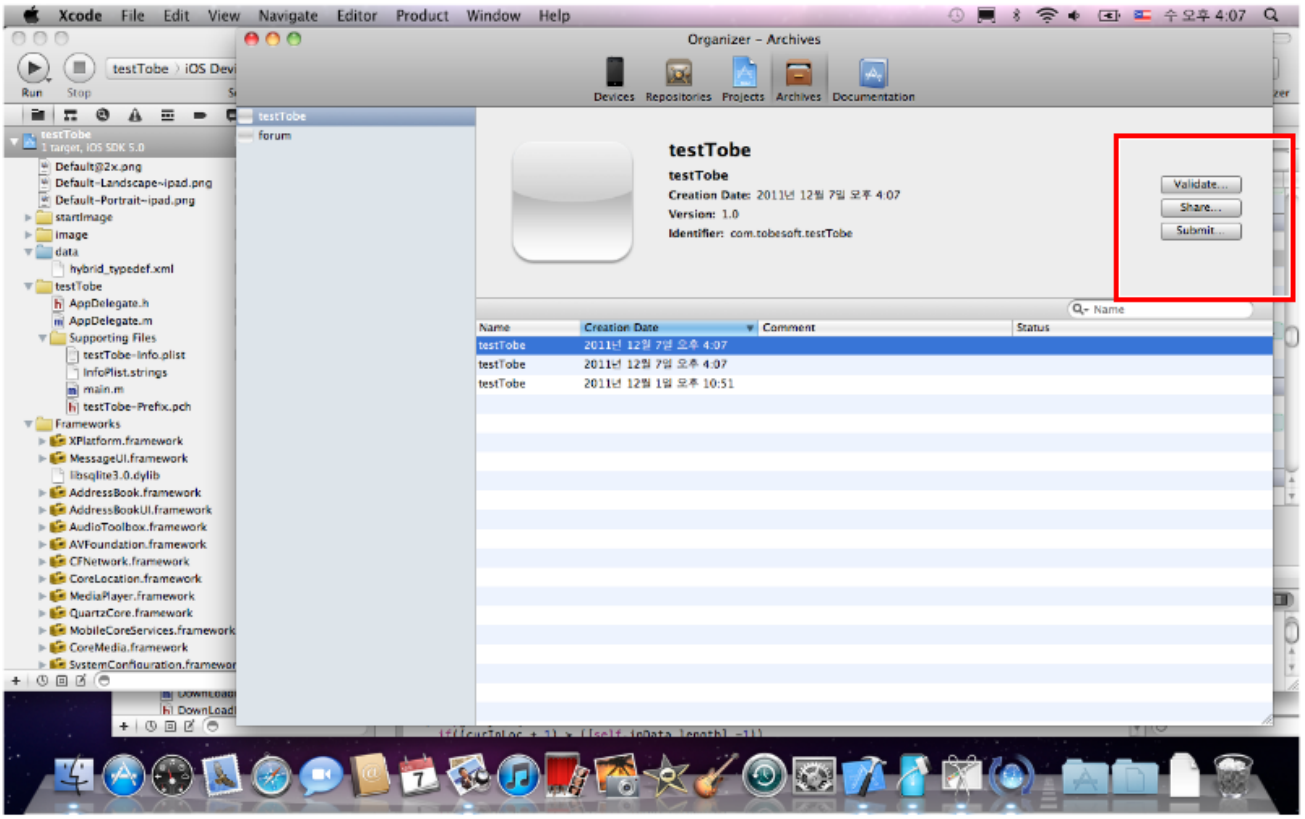
You can distribute iOS apps using the Apple iOS developer account you created when configuring the developer environment:

1. Enter app information for the distribution in **Manage Your Applications** of itunes connect
   http://itunesconnect.apple.com
2. Create **Distribution provision** for the distribution. It can be created using the developer webpage's iOS Provisioning Portal, referring to "how to".
   http://developer.apple.com/ios/manage/provisioningprofiles/viewDistributionProfiles.action
3. Build apps using the distribution provision on XCode's **Product > Archive** menu.
   After selecting the **Archive** menu, you will see a screen (see the picture below). In this picture, **Validat** checks if requirements are fulfilled for **Submit**, and a network connection is needed at this point. **Share** checks apps or test apps on other devices before the distribution. Click **Submit** and follow the process.



For XCode 4.3 and later, only the *Distribute* button is provided instead of *Share* and *Submit* buttons. The share function is included in the process followed by clicking the **Distribute** button.

4. You can distribute apps created on the *Archives* tab on Appstore XCode using the *Window* > *Organizer* menu.
5. You can check distributed apps on *Manage Your Applications* of itunesconnect.
6. You can check Apple's inspection status on the itunesconnect webpage, and the apps will be distributed to the Appstore after the inspection is completed.


Xcode File Edit View Navigate Editor Product Window Help
Organizer – Archives
Devices Repositories Projects Archives Documentation
testTobe
forum
testTobe
testTobe
Creation Date: 2011년 12월 7일 오후 4:07
Version: 1.0
Identifier: com.tobesoft.testTobe
Validate...
Share...
Submit...
Name
Creation Date
Comment
Status
testTobe 2011년 12월 7일 오후 4:07
testTobe 2011년 12월 7일 오후 4:07
testTobe 2011년 12월 1일 오후 10:51
Default@2x.png
Default-Landscape~ipad.png
Default-Portrait~ipad.png
startImage
image
data
hybrid_typedef.xml
testTobe
AppDelegate.h
AppDelegate.m
Supporting Files
testTobe-Info.plist
InfoPlist.strings
main.m
testTobe-Prefix.pch
Frameworks
XPlatform.framework
MessageUI.framework
libsqlite3.0.dylib
AddressBook.framework
AddressBookUI.framework
AudioToolbox.framework
AVFoundation.framework
CFNetwork.framework
CoreLocation.framework
MediaPlayer.framework
QuartzCore.framework
MobileCoreServices.framework
CoreMedia.framework

Part IV.

Reference

10.
Generating Code

10.1 Generating Code

Applications developed using Nexacro Platform are based on HTML5. The applications are divided into two types by operating environments: Runtime and HTML5. The Runtime version runs applications on Nexacro Platform Runner, and the HTML5 version runs applications on a web browser.

When you run Nexacro Platform applications on Nexacro Studio, you will go through a process to generate code. This process is automatically started when executing QuickView or creating archive files. You can explicitly work on any forms or any projects using the Build menu.

Internally, *nexacrogenerator.exe* converts files such as *.xadl and *.xfdl to JavaScript files.

In the distribution and operation stages that follow development and a series of tests, files sent to users need to be minimized and, in some cases, be obfuscated. Those functions can be performed through the options provided by *nexacrocompressor.exe*.

The generated script files are stored on the client. If necessary, the files can be moved to another location to be executed on the web server. If the web server is running on the client, you can designate the local web server path on Nexacro Studio to access generated files.

The following table lists the (target) files and resulting converted (code-generated) files in this process

Target File	Generated File	Description
	index.html	Basic HTML file to execute applications
	QuickView.html	HTML file to run QuickView
*.xadl	*.xadl.js	ADL file
	*.xadl.quickview.js	ADL file to run QuickView
/Service/*.xfdl	/Service/*.xfdl.js	FDL(screen) file

Target File	Generated File	Description
default.xtheme	_theme_/default/theme.css _theme_/default/theme.css.js _theme_/default/images	Theme and image file folder

10.2 Using nexacrogenerator.exe

nexacrogenerator.exe can be directly used in a console with the options described in the following table.

Options

```
nexacrogenerator.exe -A -O [args...]
```

Option	Description
-A <Target ADL Path>	Specifiy the targeted ADL path of Application to generate. This option must be entered.
-O <Output Path>	Specify the output path where generate files are stored. Automatically generate a route if the specified path does not exist. This option must be entered.
-B <library Path>	Specify the path to the library files needed to generate files. If the path is not entered, the default library files in the path where "nexacro.exe" is installed are used.
-P <Project Name>	Specify the name of a project where generated files are saved. The project name will be added to the output directory to serve as a path for saving files. If a project name is not entered, the XADL file name entered in the target ADL path is used automatically.
-AO	Only used to generate XADL files Can be changed to the application(ADL) path of the nexacro executable files Note : Can not be used in combination with -S, -F, -T.
-S <Service ID>	Specify the Service Prefix ID to generate. Can specify only Service Prefix IDs defined in TypeDefinition Note : Can not be used in combination with -AO, -F, -T.
-F <path1>,[<path2>,…]	Specify a list of paths of files to generate. Separated by a comma or semicolon when multiple inputs exist Note : Can not be used in combination with -AO, -S, -T.

Option	Description
-T <themeid1>,[<themeid2>,…]	Specify the Theme ID to generate. Separated by a comma or semicolon when multiple inputs exist Note : Can not be used in combination with -AO, -S, -F.
-TABORDER	Generate components according to their tab order. In not designated, components will be generated according to Z-order.
-L <Path>	Specify the log file path to save the generator process. Automatically generate a route if the specified path does not exist. Creating a log file may increase generator processing time.
-H	Provide the help for "nexacrogenerator.exe".

The -H option supports Korean, English, and Japanese depending on user system settings.

Examples

Generating Entire Application

Generate an application based on *C:\TestGenerate\TestGenerate.xadl* file and output the result to *E:\ResultGenerate*.
Also generate executable files and theme files required to run the application.

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate"
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
...
```

Assigning Output Path User-specified Name Instead of XADL Name

Change the project path name to "TestGenerate2" and output the result to *E:\ResultGenerate* when generating an application based on *C:\TestGenerate\TestGenerate.xadl* file.
Also generate executable files and theme files required to run the application.

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -P "TestGenerate2"
```

```
E:\ResultGenerate\TestGenerate2\index.html
E:\ResultGenerate\TestGenerate2\QuickView.html
E:\ResultGenerate\TestGenerate2\popup.html
E:\ResultGenerate\TestGenerate2\start.json
E:\ResultGenerate\TestGenerate2\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate2\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate2\_theme\…
E:\ResultGenerate\TestGenerate2\Base\…
…
```

Specifying Separate Library Path

Designate *C:\nexacro14lib* as a library path and output the result to *E:\ResultGenerate* when generating an application based on *C:\TestGenerate\TestGenerate.xadl* file.
If a library path is not specified, the default library files in the path where "nexacro.exe" is installed are used .

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -B "C:\nexacro14lib"
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
…
```

Generating XADL File Independently

Generate only *C:\TestGenerate\TestGenerate.xadl* file and output the result to *E:\ResultGenerate*.
Also generate executable files required to run the application, but not theme files.
-T, -S, -F, -AO options cannot be used simultaneously.

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -AO
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
```

Generating Specific Service Path

Generate a Service Prefix ID defined in TypeDefinition based on *C:\TestGenerate\TestGenerate.xadl* file and output the result to *E:\ResultGenerate*.
Only one service ID can be entered.
The input value for the -S option must agree with the relevant Service Prefix ID defined in TypeDefinition.
-T, -S, -F, -AO options cannot be used simultaneously.

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -S "Base"
```

```
E:\ResultGenerate\TestGenerate\Base\…
E:\ResultGenerate\TestGenerate\Base\…
…
```

Generating Specific Files

Generate *C:\TestGenerate\Base\form1.xfdl* and *C:\TestGenerate\Base\form2.xfdl* files based on *C:\TestGenerate\TestGenerate.xadl* file and output the result to *E:\ResultGenerate*.
One or multiple files can be entered.
However, XADL and XTHEME files cannot be input values for the -F option.
-T, -S, -F, -AO options cannot be used simultaneously.

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -F "C:\TestGenerate\Base\form1.xfdl",
"C:\TestGenerate\Base\form2.xfdl"
```

```
E:\ResultGenerate\TestGenerate\Base\form1.xfdl.js
E:\ResultGenerate\TestGenerate\Base\form2.xfdl.js
```

Generating Specific Theme

case 1) Theme registered in the cache folder

Generate *%USERAPP%\Cache\Theme\Gray.xtheme* file based on *C:\TestGenerate\TestGenerate.xadl* file and output the result to *E:\ResultGenerate*
When generating a theme registered in the cache folder, enter just a theme name such as "Gray.xtheme." (i.e. The generator recognizes the input as a theme registered in the cache folder when just a theme name is entered without specifying a relative path.)

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -T "Gray.xtheme"
```

```
E:\ResultGenerate\TestGenerate\_theme_\Gray\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Gray\images\...
…
```

case 2) Theme registered in a relative path or Prefix path, based on an XADL file

Generate *C:\TestGenerate\Themes\NewTheme1.xtheme* file based on C:\TestGenerate\TestGenerate.xadl file and output the result to *E:\ResultGenerate*.
Enter a relative path or Prefix path based on an XADL file in order to generate a theme in an user-specified path

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -T "Themes\NewTheme1.xtheme"
```

```
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\images\...
…
```

case 3) Combination of case 1 and case 2

Generate *%USERAPP%\Cache\Theme\Gray.xtheme* file and *C:\TestGenerate\Themes\NewTheme1.xtheme* file based on *C:\TestGenerate\TestGenerate.xadl* file and output the result to *E:\ResultGenerate*.

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -T "Gray.xtheme,Themes\NewTheme1.xtheme"
```

```
E:\ResultGenerate\TestGenerate\_theme_\Gray\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Gray\images\...
…
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\theme.css.js
E:\ResultGenerate\TestGenerate\_theme_\Themes\NewTheme1\images\...
…
```

Saving Generation Result as Log File

Generate an application based on *C:\TestGenerate\TestGenerate.xadl* file and output the result to *E:\ResultGenerate*. Then, save the details of the generation in the form of *E:\ResultGenerate\Log\TestGenerate_1.1.01.txt* file.

```
nexacrogenerator.exe -A "C:\TestGenerate\TestGenerate.xadl" -O
"E:\ResultGenerate" -L "E:\ResultGenerate\Log\TestGenerate_1.1.01.txt"
```

```
E:\ResultGenerate\TestGenerate\index.html
E:\ResultGenerate\TestGenerate\QuickView.html
E:\ResultGenerate\TestGenerate\popup.html
E:\ResultGenerate\TestGenerate\start.json
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.js
E:\ResultGenerate\TestGenerate\TestGenerate.xadl.quickview.js
E:\ResultGenerate\TestGenerate\_theme\…
E:\ResultGenerate\TestGenerate\Base\…
…
E:\ResultGenerate\Log\TestGenerate_1.1.01.txt
```

10.3 nexacrocompressor.exe

nexacrocompressor.exe can be directly executed in the console by using the options described in the below table.

Options

```
nexacrocompressor.exe -F -O [args...]
nexacrocompressor.exe -P -O [args...]
```

Option	Description
'-F <file1>,[<file2>,…]	Specify the path of files to compress. Separated by a comma or semicolon when multiple inputs exist -F or -P option must be entered, but both options can not overlap. Note: -F and -O must have the same order and number of files.
-P <directory>	Specify a target directory to compress. -F or -P option must be entered, but both options can not overlap. Note : Only one directory must be specified for -O and -P respectively.
-O <file1>,[<file2>,…] -O <directory>	-F used Specify the output path to save compressed files. Separated by a comma or semicolon when multiple inputs exist Note : If -F is used, -F and -O must have the same order and number of files. -P used Specify the output path to save compressed files. Note : If -P is used, only one directory must be specified for -O and -P respectively. Automatically generate a route if the specified path does not exist. This option must be entered. Note : If -F is used, -F and -O must have the same order and number of files.
-S <string>	Specify the suffix in the name of a compressed file.
-Shrink	Obfuscate a variable name. The eval function is included in the process of obfuscating a variable name.
-IgnoreEval	Exclude the eval function in the process of obfuscating a variable name. Note: To use this option, the -Shrink option must be used first.
-H	Provide the help for "nexacrocompressor.exe".

Examples

Compressing Specific JS Files

Compress *C:\JS\test1.js* and *C:\JS\childdir\test3.js* files and output them as *E:\JS\test1.js* and *E:\JS\childdir\test3.js* respectively.
Footnotes and spaces are removed.

```
nexacrocompressor.exe -F "C:\JS\test1.js,C:\JS\childdir\test3.js" -O
"E:\JS\test1.js,E:\JS\childdir\test3.js"
```

```
E:\JS\test1.js
E:\JS\childdir\test3.js
```

Compressing JS Files in Specific Directory

Compress the JS files in *C:\JS* directory and output them to *E:\JS* directory.
Footnotes and spaces are removed.

```
nexacrocompressor.exe -P "C:\JS" -O "E:\JS"
```

```
E:\JS\test.js
E:\JS\test1.js
E:\JS\test2.js
E:\JS\childdir\test3.js
E:\JS\childdir\test4.js
```

Specifying Suffix in Compressed File's Name.

Add the suffix "_2014100601" when compressing *C:\JS\test1.js* file and outputting it as *E:\JS\test1.js*.
Footnotes and spaces are removed.

```
nexacrocompressor.exe -F "C:\JS\test1.js" -O "E:\JS\test1.js" -S "_2014100601"
```

```
E:\JS\test1_2014100601.js
```

Obfuscating Variables for Compressed File

Obfuscate variables when compressing *C:\JS\test1.js* file and outputting it as *E:\JS\test1.js*.
Footnotes and spaces are removed.
Obfuscate all the variables both inside and outside the eval function.

```
nexacrocompressor.exe -F "C:\JS\test1.js" -O "E:\JS\test1.js" -Shrink
```

```
E:\JS\test1.js
```

Obfuscating Variables outside eval Function

Obfuscate variables except for the ones inside the eval function when compressing *C:\JS\test1.js* file and outputting it as *E:\JS\test1.js*.
Footnotes and spaces are removed.
Obfuscate the variables outside the eval function only.

```
nexacrocompressor.exe -F "C:\JS\test1.js" -O "E:\JS\test1.js" -Shrink
-IgnoreEval
```

```
E:\JS\test1.js
```

11.

Dataset XML Format

The Nexacro Platform provides X-API as a server module. Using this Nexacro Platform API, Runtime interprets the created dataset format and displays applications on the UI screen.

This chapter explains the dataset format created by X-API. XML files abiding by this format can be acknowledged as applicable data on the Nexacro Platform Runtime. It is possible to develop server programs without using X-API.

11.1 Dataset XML Layout

XML Declaration					
Root					
	Cache				
	Parameters				
		Parameter (repeat)			
	Dataset (repeat)				
		ColumnInfo			
			ConstColumn (repeat)		
			Column (repeat)		
		Rows			
			Row (repeat)		
				Col (repeat)	
				OrgRow	
					Col (repeat)

XML Declaration

The XML declaration is defined as follows.

```
<?xml version="1.0" encoding="utf-8"?>
```

The XML declaration should be placed at the beginning of an xml document. There should be no letter, including blanks, before the declaration.

XML Example

```
<?xml version="1.0" encoding="utf-8"?>
<Root xmlns="http://www.nexacroplatform.com/platform/dataset" ver="4000" >
     <Parameters>
          <Parameter id="service">stock</Parameter>
          <Parameter id="method">search</Parameter>
     </Parameters>
     <Dataset id="output">
          <ColumnInfo>
               <ConstColumn id="market" size="10" type="STRING" value="kse"/>
               <ConstColumn id="openprice" size="10" type="INT"
               value="15000"/>
               <Column id="stockCode" size="5" type="STRING"/>
               <Column id="currentprice" size="10" type="INT"/>
          </ColumnInfo>
          <Rows>
               <Row>
                    <Col id="currentCode">10001</Col>
                    <Col id="currentprice">5700</Col>
               </Row>
               <Row>
                    <Col id="currentCode">10002</Col>
                    <Col id="currentprice">14500</Col>
               </Row>
          </Rows>
     </Dataset>
</Root>
```

11.2 Dataset Elements

The dataset elements are described here.

Root

Outline

The top-level element that indicates a dataset.

Child element

Parameters, dataset

Repeat

One

Property

```
<Root xmlns="http://www.nexacroplatform.com/platform/dataset">
<Root xmlns:nexacro="http://www.nexacroplatform.com/platform/dataset">
<Root xmlns=".." version="1000">
```

Property name	Description
xmlns	Name space, if not using a basic name space then show as xmlns:nexacro
ver	Clarifies dataset layout version.

Parameters

Outline

An element clarifying the parameters needed for communication. The "child parameter" has the actual value, and the "parameters" element is an assemblage of (child) parameters .

Child element

Parameter

Repeat

None or one

Parameters > Parameter

Outline

Clarifies the value of a parameter

Child element

None

Repeat

None or as many as the number of parameters

Property

```
<Parameter id="ErrorMsg" type="STRING">SUCC</ Parameter >
```

Property Name	Description
id	Parameter name
type	Parameter type

Note

The Nexacro Platform uses the required parameters described below. You need to set parameters for the interconnection with Nexacro Platform.

```
<Parameter id="ErrorCode">1</Parameter>
<Parameter id="ErrorMsg">SUCC</Parameter>
<Parameter id="cachetype">Session</Parameter>
```

ID

ID	Description
ErrorCode	Error Code ● When calling the transaction function: ○ In the case it is less than 0, which is a user defined-error, it will be processed as "fail" without reflecting UpdateStatus into InputDataset ○ In the case it is greater than 0, which is a user defined-normal status, it will be processed as "sucess" as reflecting InputDataset into UpdateStatus ● When the load function is called: ○ In the case it is less than 0, it is processed as fail defined by the user. ○ In the case it is greater than 0, it is is processed as success as normal defined by the user.

ID	Description
	● In case not designated : 0
ErrorMsg	Error Message When ErrorCode is not specified or 0, it is success. Otherwise, fail
CacheType	Defines the cache method type: ● Type is dynamic, static or none ● In case not designated : Session

Type

The data type can be one of the following.

- STRING: Regardless of the column size, there is no limit to the width. (However it is limited to about 2GB since the system can allot 2GB at a time)
  - In the case of 0 length string
    - <Parameter id="ErrorMsg" type=”STRING”></Parameter>
  - In the case of null
    - Is divided by <Parameter id="ErrorMsg" type=”STRING”/>
- INT: Whole number(-2^{31} ~ $2^{31}-1$)
- FLOAT, DECIMAL: ($\pm 2.2X10^{-308}$ ~$\pm 1.7X10^{308}$)
- BIGDECIMAL: Real number($\pm 10^{-1056}$ ~ $\pm 10^{1056}$)
- DATE: YYYYMMDD(-8192/01/01 ~ 8191/12/31)
- DATETIME : YYYYMMDDHHmmssuuu (Possible to include msec)
- TIME : 6 digits HHmmssuuu(Possible to include msec)
- BLOB: Binary data

Dataset

Outline

Element that has the value of one dataset. A dataset has *ColumnInfo* which defines its structure and *Rows* which has the actual data as child elements.

Child element

ColumnInfo, Rows

Repeat

None or as many as the number of dataset XML passes

Property

```
<Dataset id="History">
```

Property Name	Description
id	Dataset name

Dataset > ColumnInfo

Outline

It is used to express the dataset schema. Child elements *ConstColumn* and *Column*- show the actual data. *ColumnInfo* is an assemblage of the *ConstColumn* and the *Column*. The *ConstColumn* shows the schema of the columns with fixed values. The *Column* shows the schema of the columns with non-fixed values.

Child element

ConstColumn, Column

Repeat

None or one

Dataset > ColumnInfo > ConstColumn

Outline

Expresses the schema as *Column* that always has a fixed value.

Child element

None

Repeat

None or as many as the number of fixed value column

Property

```
<ConstColumn id="systemName"
     size="255"
     type="STRING"
     value="Production"/>
```

Property Name	Description
id	Column name
size	Maximum data size
type	Data type of column
value	Fixed value of column

Note

Following are the data type that can come in type.

- STRING: Regardless of the column size there is no limit to the width. (However is limited to around 2GB which is the size the system can allot at once)
- INT: Whole number(-2^{31} ~ $2^{31}-1$)
- FLOAT, DECIMAL: ($\pm 2.2X10^{-308}$ ~$\pm 1.7X10^{308}$)
- BIGDECIMAL: Real number($\pm 10^{-1056}$ ~ $\pm 10^{1056}$)
- DATE: YYYYMMDD(-8192/01/01 ~ 8191/12/31)
- DATETIME : YYYYMMDDHHmmssuuu (Possible to include msec)
- TIME : 6 digits HHmmssuuu(Possible to include msec)
- BLOB: Second echelon data

Dataset > ColumnInfo > Column

Outline

It clarifies the value of a parameter

Child element

None

Repeat

None or as many as the number of non-fixed value columns

Property

```
<Column id="department"
     size="255"
     type="STRING"
     prop="text"
     sumtext="total"/>
```

Property Name	Description
id	Column name
size	Maximum data size
type	Data type of the column
Prop	Defines the summary type
Sumtext	Character string when prop value is text

Note

The value set in the *type* will be the same as the type in *ColumnInfo*.

The *prop* has the following values.

- count: The number of rows of the column
- sum: The sum of row values of the column
- max: The maximum value of the rows in the column
- min: The minimum value of the rows of the column
- avg: The average of the row value of the column
- Text: Sumtext property value

Dataset > Rows

Outline

It is an assemblage of the rows of a dataset. It has a row that shows the actual data as its child element.

Child element

Row

Repeat

None or one

Dataset > Rows > Row

Outline

It expresses the data of a row in the dataset. It has *Col* as its child element, which expresses value of each column. When the type is "update", it has another child element *Org_Row* that has the original values of a row.

Child element

Col, Org_Row

Repeat

None or as many as the number of rows of dataset

Property

```
<Row type="insert">
     <Col id="currentCode">10001</Col>
     <Col id="currentprice">13400</Col>
</Row>

<Rows>
     <Row type="update">
          <Col id="currentCode">10001</Col>
          <Col id="currentprice">13400</Col>
          <OrgRow>
               <Col id="currentCode">10001</Col>
               <Col id="currentprice">13700</Col>
          </OrgRow>
     </Row>
</Rows>

<Row type="delete">
     <Col id="currentCode">10001</Col>
     <Col id="currentprice">13400</Col>
</Row>
```

Property Name	Description
type	insert : A row added to the original dataset
	update: A row modified in the original dataset. It includes Org_Row as its child element. Org_Row is the original row.

Property Name	Description
	delete: A row deleted from the original dataset

Dataset > Rows > Row > Col

Outline

It expresses the value of each column of a dataset

Child element

None.

Repeat

None or as many as the number of columns

Property

```
<Col id="department">management</Col>
```

Property Name	Description
Id	Column name. It Is the same as the name set in the child elements *(ConstColumn, Column)* of *ColumnInfo*

Note

In the case of a 0-length string:

```
<Col id="department"></Col>
or
<Col id="department"/>
```

In the case of null, it is classified as without "tag".

Dataset > Rows > Row > OrgRow

Outline

It has the original value when the value of dataset's row is modified. It has Col which has the actual value as its child element.

Child element

Col

Repeat

None or one.

Note

Similar to Row which is its parent element but has no property

Dataset > Rows > Row > OrgRow > Col

Outline

It expresses the original column value before the value is modified

Child element

None

Repeat

None or as many as the number of columns

Property

```
<Col id="department">management</Col>
```

Property Name	Description
id	Column name. It is the same as the name set in the child elements *(ConstColumn, Column)* of *ColumnInfo*

Appendix A.
Setting Path

A.1 Alias Path

Nexacro Platform uses the following alias paths for setting paths on the user PC.

Alias Path Name	Actual Path Name
%nexacro%	%PROGRAMFILES%\nexacro\14
%USERAPP%	%LOCALAPPDATA%LOW\nexacro\14 \data\data\[packageName]\files\nexacro\ (Android) \Library\Caches\nexacro\ (iOS)
%UPDATE%	%USERAPP%\Update
%CACHE%	%USERAPP%\Cache
%WEBDEPLOY SETUP%	Server Context root
%WEBDEPLOY PROJECT%	Server Context root
%WEBDEPLOY FRAMEWORK%	%WEBDEPLOY PROJECT%\nexacro14lib\framework
%WEBDEPLOY COMPONENT%	%WEBDEPLOY PROJECT%\nexacro14lib\component
%WEBDEPLOY RESOURCE%	%WEBDEPLOY PROJECT%\nexacro14lib\resources
%WEBDEPLOY THEME%	%WEBDEPLOY PROJECT%_theme_

This alias path can be used when inputting distribution information paths or path information in application programs.

A.2 nexacro.xml

The *nexacro.xml* file is XML used to contain information for each user. The file path is as follows.

> %USERAPP%

A.3 Relative Path

Relative Path Support for Paths in Projects

The chart below shows the types of major paths and relative path support, and service usage path support (Service::a.xfdl format path).

File Type	Category	Relative Path		Supports Service Usage Path
		Support	Standard Path	
Typedefinition	Component UpdateURL ❶	Y	TypeDefintion	Y
	Service Path	Y	TypeDefintion	N
GlobalVariable	Image	Y	GlobalVariable	Y
XADL	TypeDefinition ❷	Y	XADL	N
	GlobalVariable	Y	XADL	Y
	EngineUrl ❸	Y	XADL	N
	LiceneceUrl ❹	Y	XADL	N
	Themeid	Y	XADL	Y

1. Component UpdateURL
   A path for the server to update components. It also corresponds to the *TypeDefinition Update Url* in tool. However, if its domain (for rexample, *http://a.b.c/path* > *a.b.c*) isn't matched with the XADL path, update is not available. If XADL has a local path, update is available regardless of the domain information.
2. TypeDefinition
   Paths for services. It also corresponds to *TypeDefinition* Services url in tool.
3. Engine path
   A path to update the engine. It also means ADL *engineurl* in tool. However, if its domain is not matched with the XADL path, update is not available. If XADL has a local path, update is available regardless of the domain information.
4. Licenece path
   A path to check the license before reading the *Typedefinition*.
   It means *ADL licenseurl* in Tool.
   However, its domain should be not matched with the XADL path. If XADL has a local path, the license will be checked regardless of the domain.

Plugin Method

The following table describes the types of paths set to HTML, as well as relative paths support, when running Nexacro Platform HTML plugIn method.

Classification (HTML)	Item	HTML Relative Path	XADL Relative Path	Service Ssage Path Support
CODEBASE	SetupEngine.cab	N/A	N/A	N/A
	nexacro14_Launcher.cab	N/A	N/A	N/A
SetupEngine	Xadlpath property	supported	N/A	N/A
NexacroLauncherAX	Xadlpath property	supported	N/A	N/A
	Loadingimage property	supported	N/A	N/A
	Splashimage property	supported	N/A	N/A
	makeshortcut()	supported	N/A	N/A

4GL doesn't support those relative paths. Only HTML supports those paths.

The standard paths for HTML relative paths are HTML paths.

Appendix B. Exceptions

This chapter is for reference only since the user environment will vary. It describes issues you may encounter while using the Nexacro Platform and presents possible solutions..

B.1 Web Browser Option

Changing web browser options may impact how an application runs as described in this section.

JavaScript Activation

Issue

Setting the web browser to not use JavaScript causes the application to not function normally.

Description

Even though basic options are set to activate JavaScript in general, you need to check the relevant option first when the applications cannot be found on the screen since you can choose other options.

Possible Solution

Depending on the web browser, activate the JavaScript usage option. (The option can vary depending on brower version.) Refer to the following link or each web browser's help for details.

```
http://www.enable-javascript.com
```

For Internet Explorer, you can change the option using the following menu:

```
Tools > Internet options > Security > Custom level > Internet Zone > Scripting
> Active scripting
```

File Download Activation

Issue

Even though you restrict to download files from web browsers, files can be downloaded using File download component.

Possible Solution

Even though basic options are set to activate file download, you need to check the relevant option first when the files are not downloaded since you can choose other options.

Depending on the browser, activate the file download option. (The option can vary depending on browser version.) Refer to the web browser's help for details.

In case of Internet Explorer, you can change the option using the following menu.

```
Tools > Internet options > Security > Custom level > Internet Zone > Downloads
> File Download
```

HTTP 1.1 Activation

Issue

If you restrictI use of HTTP 1.1 on the web browser, applications might work slowly or might not work at all.

Possible Solution

Even though basic options are set to activate HTTP 1.1, you need to check the relevant option first when the applications do not work, since you can choose other options.

This option is provided in Internet Explorer only, and can be changed using the following menu.

```
Tools > Internet options > Advanced > HTTP settings > Use HTTP 1.1 through
proxy connections
Tools > Internet options > Advanced > HTTP settings > Use HTTP 1.1
```

When you set a web server to use HTTP 1.0 regardless of web browser settings, an application might work slowly or might not work at all. You need to check the web server options if the entire user environment gets slow.

In case of Apache server, you can use `force-response-1.0` setting.

http://httpd.apache.org/docs/2.2/en/env.html#special

XMLHTTP Activation

Issue

If you restrict to use XMLHTTP on a web browser, applications cannot work actively.

Possible Solution

Even though basic options are set to activate XMLHTTP, you need to check the relevant option first when the application does nt work since you can choose other options.

In Internet Explorer you can change the option using the following menu.

```
Tools > Internet options > Advanced > Security > Enable native XMLHTTP support
```

B.2 Internet Explorer Compatibility View

Issue

If you add a domain serviced on Internet Explorer as a target for compatibility view mode, the screen might not be displayed normally.

Possible Solution

An HTML file to be used when developing Nexacro Platform applications is set in rendering mode by the Meta tag, so it can work in Internet Explorer's standard mode(IE=Edge) to utilize improved features of later versions. Consequently, the compatibility view icon is not shown in the address window in the corresponding mode.

For this reason, a guide for standard mode operation can be provided after checking the user's browser settings, or the user can be informed of the compatibility view mode with a separate script.

The codes below are for reference only. You need to adjust the codes depending on the development environment.

Note: http://msdn.microsoft.com/en-us/library/ms537503(v=vs.85).aspx

```
var agentStr = navigator.userAgent;

var mode;
if (agentStr.indexOf("Trident/6.0") > -1) {
     if (agentStr.indexOf("MSIE 7.0") > -1) {
```

```
          mode = "IE10 Compatibility View";
     } else {
          mode = "IE10";

}
else if (agentStr.indexOf("Trident/5.0") > -1) {
     if (agentStr.indexOf("MSIE 7.0") > -1) {
          mode = "IE9 Compatibility View";
     } else {
          mode = "IE9";

}
else if (agentStr.indexOf("Trident/4.0") > -1) {
     if (agentStr.indexOf("MSIE 7.0") > -1) {
          mode = "IE8 Compatibility View";
     } else {
          mode = "IE8";
     }
}
else {
     mode = "IE7";
}

document.title = "Browser Mode:\t" + mode;
```

B.3 Adding Web Content Using Iframe

Issue

If Nexacro Platform contents are added in the form of iframe on the existing web screen which is already developed and operated, the screen might not be displayed on Internet Explorer.

Possible Solution

An HTML file to be used when developing Nexacro Platform applications is configured to work in Edge mode. If the HTML file used on the existing screen is not set as DTD (Document Type Definition), it works in Quirks mode. In this process, the screen might not be displayed normally.

In this case, add the Meta tag as shown below.

```
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
</head>
```

Refer to the following link for compatibility guarantee using the Meta tag or HTTP header.
http://technet.microsoft.com/ko-kr/library/gg699448.aspx

Appendix C.
Installing Nexacro Platform X-APIs

Nexacro Platform APIs (X-APIs) are provided in the form of server modules, and serve the data processing between a server and clients. X-APIs simplify data processing by providing functions such as sending, receiving, and processing data.

Data is processed according to the 11. Dataset XML Format Therefore, users can develop server programs without using X-APIs.

C.1 Installation

X-APIs, provided in the form of JavaScript-based server modules, require JDK or JRE (version 1.4 or higher).

Users will copy the JAR files included in the *nexacro-xapi/lib* directory of the installation CD and move them to the *WEB-INF/lib* directory of the WAS or to defined class paths. The offered JAR files are as follows.

File name	Necessity	Description	Reference
nexacro-xapi-x.x.x.jar	Y	X-API	
commons-logging-x.x.x.jar	Y	X-API internal logging	Apache Commons Logging

C.2 License Authentication

Users will copy the license file (*nexacro14_server_license.xml*) to the same directory or defined class path as the one that contains JAR files.

If one or more other license files are copied to different paths, the license file located in the same directory with JAR files is first to be applied.

Depending on the setup of the server, the WAS requires re-booting after JAR files and a license file are copied.

C.3 Confirming Installation

Whether the installation was successful can be confirmed by writing the sample JSP file presented below.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
    <head>
    <title>JarInfo</title>
    <style>
        * { font-family: Verdana }
    </style>
    </head>
    <body>
        <pre>
<%
    new com.nexacro.xapi.util.JarInfo().info(out);
%>
        </pre>
    </body>
</html>
```

If successful, details of the X-API version and product license will be output.

```
 --- Manifest ---
Manifest-Version: 1.0
Built-By: Server Team
Created-By: 1.4.2_19-b04 (Sun Microsystems Inc.)
Ant-Version: Apache Ant 1.7.1
Main-Class: com.nexacro.xapi.util.JarInfo
Built-Date: April 23 2014

Name: com.nexacro.xapi
Specification-Title: X-API
Implementation-Title: X-API
Specification-Version: 1.0
Specification-Vendor: [redacted]
Implementation-Vendor-Id: com.nexacro
Implementation-Version: 1.0.0.4017
Implementation-Vendor: [redacted]

--- Log ---
INFO Loaded license file in JAR dir: path=D:\workspace\nexacro\WebContent\WEB-INF\lib\nexacro14_server_lic

--- License ---
product.name: nexacro platform
product.version: 14
product.function: Windows, WindowsRT, MAC, iOS, Android
customer.name: [redacted]
server.cpuCoreCount: [redacted]
server.ipAddress: [redacted]
date.activation: 2014-04-01
date.term: 2
date.term.unit: month
```

Appendix D.
Installing Nexacro-Xeni

Nexacro-Xeni is a server module that provides necessary features for clients to process data in file forms. It is designed to transfer data from a file to a grid or vice versa.

Nexacro-Xeni supports the following file formats and features.

File format	Extension	Features
Microsoft Excel 97-2003	xls	export / import
Microsoft Excel 2007/2010	xlsx	export / import
CSV format file	csv	import

D.1 Installation

Java-based Nexacro-Xeni requires JDK or JRE (1.4 or later version).

You can deploy the provided Web application ARchive (WAR) file. Otherwise, you can unzip the WAR file in your PC and then copy necessary files to the **/WEB-INF/lib** directory or to the defined class path. The below list presents main Java Archive (JAR) files included in the WAR file.

```
commons-codec-1.5.jar
commons-fileupload-1.3.1.jar
commons-io-2.2.jar
commons-logging-1.1.jar
dom4j-1.6.1.jar
log4j-1.2.13.jar
nexacro-xapi-1.0.jar
nexacro-xeni-1.1.jar
poi-3.9-20121203.jar
poi-ooxml-3.9-20121203.jar
poi-ooxml-schemas-3.9-20121203.jar
```

```
poi-scratchpad-3.9-20121203.jar
stax-api-1.0.1.jar
xmlbeans-2.3.0.jar
xmlpull-1.1.3.1.jar
xpp3_min-1.1.4c.jar
xstream-1.4.2.jar
```

Copying the above files to a server rather than deploying the WAR file may cause problems due to the disparity in library versions.

Nexacro-Xeni utilizes the Apache POI library. Click the following link for the details on Apache POI.
https://poi.apache.org/

To run Nexacro-Xeni, you must install X-APIs. To that end, please refer to Installing Nexacro Platform X-APIs.

If you copy JAR files to the existing context, you must add the below codes to the *web.xml* file.

```
<servlet>
    <servlet-name>XExportImport</servlet-name>
    <servlet-class>com.nexacro.xeni.services.GridExportImportServlet</servlet-c
lass>
</servlet>
<servlet>
    <servlet-name>XImport</servlet-name>
    <servlet-class>com.nexacro.xeni.services.GridExportImportServlet</servlet-c
lass>
</servlet>

<servlet-mapping>
    <servlet-name>XExportImport</servlet-name>
    <url-pattern>/XExportImport</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>XImport</servlet-name>
    <url-pattern>/XImport</url-pattern>
</servlet-mapping>
```

```
<context-param>
    <param-name>export-path</param-name>
    <param-value>/export</param-value>
</context-param>
<context-param>
    <param-name>import-path</param-name>
    <param-value>/import</param-value>
</context-param>
<context-param>
    <param-name>monitor-enabled</param-name>
    <param-value>true</param-value>
</context-param>
<context-param>
    <param-name>monitor-cycle-time</param-name>
    <param-value>30</param-value>
</context-param>
<context-param>
    <param-name>file-storage-time</param-name>
    <param-value>10</param-value>
</context-param>
```

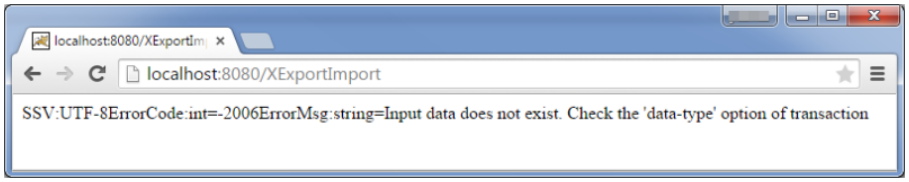
The JAR file list and codes of *web.xml* as described above represents the 1.1 version of Nexacro-Xeni. Those contents may change later in accordance with future updates.

D.2 Confirming Installation

To confirm the successful installation, open the below URL through a web browser.

```
http://127.0.0.1:8080/XExportImport
```

Then, check that the below message is returned.


localhost:8080/XExportIm
localhost:8080/XExportImport
SSV:UTF-8ErrorCode:int=-2006ErrorMsg:string=Input data does not exist. Check the 'data-type' option of transaction

The URL differs according to installation paths. If you deploy the WAR file rather than copying the inside files, the below URL will be given to you.
`http://127.0.0.1:8080/nexacro-xeni/XExportImport`

D.3 Settings

The below table presents the main parameters registered in the servlet context.

Name	Default value	Description
export-path	/export	Designates the path where downloaded files are temporarily saved.
import-path	/import	Designates the path to hold temporary files whose data is processed into grid components. If the value of the *monitor-enabled* parameter is set to "true," the temporary files will be deleted right after the data processing.
monitor-enabled	true	Determines whether to run the file manager. If "false," the file manager will not operate while the temporary files will not be deleted.
monitor-cycle-time	30	Sets the interval for deleting temporary downloaded files by minute. At determined intervals, the file manager will operate, check the files that are subject to deletion, and finally remove those files.
file-stroage-time	10	Sets the period for storing temporary downloaded files by minute. When the file manager operates, it deletes the files that have been stored longer than a determined period.

For further details, please refer to the *readme.txt* file included in the WAR file.

D.4 Example

This section explains simple examples of processing data using Nexacro-Xeni. Go to the below item in the help menu for the details on properties, methods and events

```
Objects > Misc. Objects > ExcelExportObject
```

Export

The process of exporting data begins with sending grid-bound data to the server. Next, the server creates a file based on the data to allow users to download the file. In the below sample script, users will download the file derived from the data bound to the Grid00 component.

```
this.Button00_onclick = function(obj:Button,  e:nexacro.ClickEventInfo)
{
    this.exportObj = new ExcelExportObject();
    var ret = this.exportObj.addExportItem(nexacro.ExportItemTypes.GRID,
        this.Grid00, "Sheet1!A1");
    this.exportObj.set_exporturl("http://localhost:8080/XExportImport");
    this.exportObj.exportData();
}
```

Import

At the beginning of the importing data process, users send their files to the server, which will convert those files into datasets. Afterward, the users will perform their tasks by downloading those datasets.

```
this.Button00_onclick = function(obj:Button,  e:nexacro.ClickEventInfo)
{
    this.importObj = new ExcelImportObject("Import00",this);
    this.importObj.addEventHandler("onsuccess", this.Import_onsuccess, this);
    this.importObj.set_importurl("http://localhost:8080/XExportImport");
    this.importObj.importData("","output=ds","Dataset00=ds");
}

this.Import_onsuccess = function(obj,e)
{
    this.Grid00.createFormat();
}
```